2021

PCIe-M60 函数手册

V1.7.5

目录

一.系统操作 4
例程 5
M_Open 6
M_Close 6
M_ResetFPGA 7
M_SetWatchdogEnable 7
M_SetWatchdogAction 8
M_WatchdogReset 8
M_GetEmg 9
M_SetEmgInv 9
M_GetEmgInv 10
M_SetEmgAction 11
M_GetEmgAction 11
M_ClrEmg 12
M_GetVersion 12
M_LoadEni 13
M_ConnectECAT 13
M_GetConnectError 14
M_DisconnectECAT 15
M_GetSlaveInfo 15
M_GetSlaveResource 16
M_WriteAxisParam 16
M_LoadParamFromFile 17
M_EnableAlias 18
M_ReadServoAlias 18
M_ReadIOAlias 19
M_ReadSlaveAlias 19
M_WriteServoAlias 20
二.轴状态指令 20
例程 22
M_Servo_On 23
M_Servo_Off 24
M_SetSoftLimit 24
M_GetSoftLimit 25
M_SetAxisBand 25
M_GetAxisBand 26
M_SetStopDec 27
M_GetStopDec 27
M_GetSts 28
M_ClrSts 29
M_GetCmd 29
M_GetCmdVel 30

M_GetEncPos 30
M_ReadActualPosition 31
M_SetCurrentPos 32
M_GetTargetPos 32
M_GetEncVel 33
M_SetAxisStsSync 33
M_GetAxisStsSync 34
M_GetEcatDigitalInput 35
M_EcatStatusWord 35
M_AxisPDOWrite 36
M_AxisPDORead 36
M_EcatSetOperationMode 37
M_EcatGetOperationMode 38
三.运动指令 39
1.绝对相对运动 39
例程 41
M_SetMove 42
M_GetMove 42
M_AbsMove 43
M_RelMove 43
2.Jog 运动 44
例程 44
M_Jog 45
3.停止运动 45
M_Stop 46
M_StopSingleAxis 46
4.回零运动 47
例程 48
M_SetHomingMode 49
M_SetHomingPrm 49
M_GetHomingPrm 50
M_HomingStart 51
M_GetEcatHomingStatus 52
M_HomeCancel 52
M_HomeCancelSingleAxis 53
5.电子齿轮运动 53
M_Gear 54
M_SetGearMaster 55
M_SetGearRatio 55

M_GearStart 56
M_GearStartSingleAxis 57
M_GetGearMaster 57
M_GetGearRatio 58

6.连续插补运动 59

例程 60
M_SetCrd 61
M_GetCrd 63
M_CrdClear 64
M_Line 64
M_Arc2R 65
M_Arc2C 66
M_BufIO 67
M_BufDelay 68
M_GetLastCrdPos 68
M_CrdStatus 69
M_CrdStart 70
M_CrdStop 71

7.速度前瞻功能 72

例程 74
M_SetVelPlanning 75
M_CrdData 76

四.IO 操作 76
M_Set_Digital_Chn_Output 78
M_Set_Digital_Port_Output 78
M_Get_Digital_Chn_Output 79
M_Get_Digital_Port_Output 80
M_Get_Digital_Chn_Input 80
M_Get_Digital_Port_Input 81
M_Set_Analog_Output 81
M_Get_Analog_Output 82
M_Get_Analog_Input 83
M_Get_Analog_Input_32 83
M_Set_Analog_Output_32 84
M_Set_Slave_Digital_Chn_Output 84
M_Set_Slave_Digital_Port_Output 85
M_Get_Slave_Digital_Chn_Output 86
M_Get_Slave_Digital_Port_Output 86
M_Get_Slave_Digital_Chn_Input 87
M_Get_Slave_Digital_Port_Input 87
M_Set_Slave_Analog_Output 88
M_Get_Slave_Analog_Output 89

M_Get_Slave_Analog_Input 89
M_Set_Slave_Analog_Output_32 90
M_Get_Slave_Analog_Input_32 91
五.报错代码 91

一.系统操作

指令	说明
M_Open	打开运动控制板卡
M_Close	关闭运动控制板卡
M_ResetFPGA	重置运动控制板卡连接芯片
M_GetEmg	获取板卡 EMG 状态
M_ClrEmg	重置板卡 EMG 状态
M_SetEmgInv	设置 EMG 信号逻辑
M_GetEmgInv	获取 EMG 信号逻辑
M_SetEmgAction	设置 EMG 触发动作
M_GetEmgAction	获取 EMG 触发动作
M_GetVersion	获取运动控制板卡版本
M_LoadEni	加载运动控制板卡 ENI 文件
M_ConnectECAT	连接 Ethercat 网络中的从站设备
M_GetConnectError	获取连接状态的错误信息
M_DisconnectECAT	断开 Ethercat 网络中的从站设备
M_GetSlaveInfo	获取 Ethercat 网络中指定从站的信息

M_GetSlaveResource	获取 Ethercat 网络中的从站资源
M_WriteAxisParam	将参数保存到文件
M_LoadParamFromFile	从文件导入参数
M_EnableAlias	开启从站别名功能
M_ReadServoAlias	获取伺服的别名数组
M_ReadIOAlias	获取 IO 从站的别名数组
M_ReadSlaveAlias	获取所有从站的别名数组
M_WriteServoAlias	设置伺服从站的别名

系统操作包含打开板卡、关闭板卡、复位板卡参数、获取 EMG 信号、重置 EMG 信号等功能。基本的注册卡片以及连接流程如下：

关闭卡片的一般流程如下：

例程

1.1 初始化卡片，然后连接总线，最后关闭卡片

```
short CardID=0;
int ret = ecat_motion.M_Open(CardID, 0); //打开卡片
string ENIPath ="C:\\Ethercat\\ENI\\eni.xml";
ret = ecat_motion.M_LoadEni(ENIPath, CardID);//加载 ENI 文件
ret = ecat_motion.M_ResetFPGA(CardID);//重启 FPGA
Thread.Sleep(500);//等待 FPGA 重启完成
ret = ecat_motion.M_ConnectECAT(0, CardID); //连接总线

.....
```

```
ret = ecat_motion.M_DisconnectECAT(CardID); //断开总线连接
ret = ecat_motion.M_Close(CardID); //关闭卡片
```

1.2 设置 EMG 为常开极性，获取 EMG 状态，并且清除状态后重新连接

```
short emg =0;
ret=ecat_motion.M_SetEmgInv(0,CardID);//设置 EMG 极性为常开
ret =ecat_motion.M_GetEmg(ref emg, CardID);
if (emg==1)
{
    ret = ecat_motion.M_ClrEmg(CardID); //重置EMG信号
}
```

M_Open

M_Open	打开运动控制板卡

指令说明: 打开运动控制板卡，在其他操作之前必须执行，同一张卡只能打开一次。

指令原型： short M_Open (short card, short param)

参数说明:

card	卡片卡号，从 0 开始，默认按主板插槽号排序
param	参数，保留

返回值 ：详见第五章。

M_Close

M_Close	关闭运动控制板卡

指令说明: 关闭运动控制板卡，在关闭程序时执行该函数。

指令原型： short M_Close (short card)

参数说明:

card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_ResetFPGA

M_ResetFPGA	重置运动控制板卡连接芯片

指令说明: 重置运动控制板卡连接芯片，连接前需要重置下连接芯片，确保本次连接能够正常执行，执行函数以后需要等待 500ms 来确保 FPGA 芯片能够正常重启。

指令原型: short M_ResetFPGA (short card)

参数说明:

card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_SetWatchdogEnable

M_SetWatchdogEnable	开启看门狗功能

指令说明: 开启看门狗功能，当开门狗被触发以后需要重新开启。可以通过看门狗功能确保程序在设定时间内无反应的情况下自动执行设定的看门狗动作，保证设备的安全性。

指令原型: short M_SetWatchdogEnable (short enable, int period, short card)

参数说明:

enable	是否开启看门狗，1-开启　0-关闭
period	喂狗周期，如果该周期内没有执行喂狗，则会触发看门狗动作 单位 ms。推荐 1000-2000
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值　：详见第五章。

M_SetWatchdogAction

M_SetWatchdogAction	设置看门狗的动作

指令说明: 设置看门狗动作，当程序在规定时间内没有执行喂狗动作时，会触发看门狗的动作。

指令原型： short M_SetWatchdogAction (short actionMask，short card)

参数说明:

actionMask	看门狗执行动作，Bit0-伺服掉使能 Bit1-平滑减速停机 Bit2=紧急减速停机
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值　：详见第五章。

M_WatchdogReset

M_WatchdogReset	设置看门狗的动作

指令说明: 喂狗函数，规定之间内不执行该函数，板卡自动执行看门狗动作

指令原型： short M_WatchdogReset (short card)

参数说明:

card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetEmg

M_GetEmg	获取板卡 EMG 状态

指令说明: 获取运动控制板卡 EMG 信号状态，急停状态下无法连接和运动。

指令原型： short M_GetEmg (ref short emg, short card = 0)

参数说明:

emg	板卡 EMG 信号状态，1：急停中（保持信号），0：未急停
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_SetEmgInv

M_SetEmgInv	设置板卡急停信号逻辑

指令说明: 设置板卡急停信号逻辑，根据硬件接线来设定板卡的急停极性，急停信号线有 24V 供电时，为常闭逻辑。急停信号线无供电时为常开逻辑。改变电平状态则会触发急停信号。

指令原型： short M_ SetEmgInv (short senseLevel，short card = 0)

参数说明：

senseLevel	急停动作： 0x00 – 常开（默认值） 0x01 – 常闭
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetEmgInv

M_GetEmgInv	获取板卡急停信号逻辑

指令说明： 获取板卡急停信号逻辑。

指令原型： short M_ SetEmgInv (short* senseLevel，short card = 0)

参数说明：

senseLevel	急停动作： 0x00 – 常开（默认值） 0x01 – 常闭
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_SetEmgAction

M_SetEmgAction	设置板卡急停动作

指令说明: 设置板卡的急停动作，当硬件急停状态和板卡设置的急停逻辑不匹配时会触发设置的急停动作，需要注意，推荐使用掉使能动作（0x00）。

指令原型： short M_SetEmgAction (short EAction，short card = 0)

参数说明:

EAction	急停动作： 0x00 - 不减速，直接掉使能（默认值） 0x01 - 以缓停减速度停机，然后掉使能 0x02 - 以急停减速度停机，然后掉使能 0x11 - 以缓停减速度停机，不掉使能 0x12 - 以急停减速度停机，不掉使能
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetEmgAction

M_GetEmgAction	设置板卡急停动作

指令说明: 设置板卡的急停动作。

指令原型： short M_GetEmgAction (ref short EAction，short card = 0)

参数说明:

EAction	急停动作： 0x00 - 不减速，直接掉使能（默认值） 0x01 - 以缓停减速度停机，然后掉使能 0x02 - 以急停减速度停机，然后掉使能 0x11 - 以缓停减速度停机，不掉使能 0x12 - 以急停减速度停机，不掉使能
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_ClrEmg

M_ClrEmg	重置板卡 EMG 状态

指令说明: 重置运动控制板卡 EMG 信号状态。当急停的硬件信号被恢复以后，板卡会依旧保留在急停状态，需要使用程序来清除急停状态。

指令原型: short M_ClrEmg (short card = 0)

参数说明:

card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetVersion

M_GetVersion	获取运动控制板卡版本

指令说明: 获取运动控制板卡版本。

指令原型： short M_GetVersion (out char pVersion, int size, short card)

参数说明：

pVersion	字符数组的首地址指针
size	数组长度
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_LoadEni

M_LoadEni	加载运动控制板卡 ENI 文件

指令说明： 加载运动控制板卡的 ENI 文件，ENI 文件为 Ethercat 配置文件，在连接总线之前需要加载该文件，文件在驱动的默认安装目录下。

指令原型： short M_LoadEni (string eniPath, short card)

参数说明：

eniPath	Eni 文件的路径地址
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_ConnectECAT

M_ConnectECAT	连接 Ethercat 网络中的从站设备

指令说明： 运动控制板卡连接 Ethercat 网络中的从站设备，该函数用于连接总

线，当连接成功以后才可以继续执行轴和 IO 的相关操作。本条函数会根据从站数量进行耗时，没一个轴类型的从站会耗时 1s 左右。

指令原型：short M_ConnectECAT (short option, short card)

参数说明：

option	设置断线输出保持，0：不保持 1：保持
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetConnectError

M_GetConnectError	获取连接异常时的错误信息

指令说明: 获取连接总线时异常信息，目前支持在 PREOP 和 SAFEOP 状态下出现问题时获取第一个问题从站。

指令原型：short M_GetConnectError (ref uint errinfo, short card)

参数说明：

errinfo	连接报错信息，格式为 0x 0014_aabb: 0014 是报错码，aa 是状态（04 是 safe，08 是 op）,bb 是出现问题的从站逻辑顺序编号从 0 开始
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_DisconnectECAT

M_DisconnectECAT	断开 Ethercat 网络中的从站设备

指令说明: 运动控制板卡断开 Ethercat 网络中的从站设备的连接。

指令原型: short M_DisconnectECAT (short card)

参数说明:

card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetSlaveInfo

M_GetSlaveInfo	获取 Ethercat 网络中指定从站的信息

指令说明: 获取 Ethercat 网络中指定从站的信息。

指令原型: short M_GetSlaveInfo (out SL_INFO pInfo, short slaveNo,

short card)

参数说明:

pInfo	从站信息结构体 SL_INFO 的指针地址
slaveNo	从站号，从 1 开始计数
card	卡片卡号，从 0 开始，默认按主板插槽号排序

SL_INFO:

变量名	Vendor ID	ProductID	RevisionNo	SlaveType	ModuleNum	ModuleId
说明	厂家会员号	产品编号	版本号	从站类型	从站的模块数	模块 ID

返回值 ：详见第五章。

M_GetSlaveResource

M_GetSlaveResource	获取 Ethercat 网络中的从站资源

指令说明: 获取 Ethercat 网络中的从站资源。

指令原型: short M_GetSlaveResource (out SL_RES pRes, short card)

参数说明:

pRes	从站资源结构体 SL_RES 的指针地址
card	卡片卡号，从 0 开始，默认按主板插槽号排序

SL_RES:

变量名	SlaveNum	AxisNum	IoMNum	DiNum	DoNum	AINum	AoNum
说明	从站数量	伺服数量	IO 模块数量	DI 数量	DO 数量	Ai 数量	Ao 数量

返回值 ：详见第五章。

M_WriteAxisParam

M_WriteAxisParam	将板卡参数写入文档

指令说明: 将板卡参数写入文档。

指令原型: void M_WriteAxisParam (string AxisNum, string ParamName, string Param, string filePath)

参数说明:

AxisNum	轴号

ParamName	参数名称，详细名称参考下表
Param	参数数值，需要保存参数的数值
filePath	文件路径，参数 INI 文件的路径

ParamName:

变量名	SPL	SML	Band	Stime	SmAcc	EmgAcc	ACC
说明	正向软极限	负向软极限	到位阈值	稳定时间	平滑减速度	急停减速度	运动加速度
变量名	DEC	SF	HomMod	HomVel1	HomVel2	HomAcc	HomOffset
说明	运动减速度	运动加加速时间	驱动器回零方式	驱动器回零速度 1	驱动器回零速度 2	驱动器回零加速度	驱动器回零偏移

返回值 ：详见第五章。

M_LoadParamFromFile

M_LoadParamFromFile	**从文件导入参数**

指令说明: 从文件导入参数。

指令原型: short M_LoadParamFromFile(string filename, short card)

参数说明:

filename	参数文件路径
card	卡号

返回值 ：详见第五章。

M_EnableAlias

M_EnableAlias	开启从站别名功能

指令说明: 开启从站别名功能。开始后从站的 ID 根据手动设定为准，轴号根据从站 ID 分配。

指令原型： short M_EnableAlias(short enable = 0, short card = 0)

参数说明:

enable	别名功能是否开启 0-关闭 1-开启
card	卡号

返回值 ：详见第五章。

M_ReadServoAlias

M_ReadServoAlias	获取伺服的别名数组

指令说明: 获取别名开启时的伺服别名数组。

指令原型： short M_ReadServoAlias(ref short pAlias, ref short pSlaveNum, short card = 0)

参数说明:

pAlias	别名数组首地址
pSlaveNum	伺服数量返回值
card	卡号

返回值 ：详见第五章。

M_ReadIOAlias

M_ReadIOAlias	获取 IO 从站的别名数组

指令说明: 获取别名开启时的 IO 从站的别名数组。

指令原型: short M_ReadIOAlias(ref short pAlias, ref short pSlaveNum, short card = 0)

参数说明:

pAlias	别名数组首地址
pSlaveNum	IO 从站数量返回值
card	卡号

返回值 ：详见第五章。

M_ReadSlaveAlias

M_ReadSlaveAlias	获取所有从站的别名数组

指令说明: 获取别名开启时的所有从站的别名数组。

指令原型: short M_ReadSlaveAlias(ref short pAlias, ref short pSlaveNum, short card = 0)

参数说明:

pAlias	别名数组首地址
pSlaveNum	所有从站数量返回值

card	卡号

返回值 ：详见第五章。

M_WriteServoAlias

M_WriteServoAlias	**设置伺服从站的别名**

指令说明: 设置当前伺服的别名，按数组方式传输，数组序列号为伺服排列顺序。

指令原型： short M_WriteServoAlias(ref short pAlias, short slaveNum, short card = 0)

参数说明:

pAlias	别名数组首地址
pSlaveNum	伺服的从站数量
card	卡号

返回值 ：详见第五章。

二.轴状态指令

指令	说明
M_Servo_On	使能操作
M_Servo_Off	去使能操作

M_SetSoftLimit	设置指定轴的软件位置限制
M_GetSoftLimit	获取指定轴的软件位置限制
M_SetAxisBand	设置指定轴的到位判断阈值
M_GetAxisBand	获取指定轴的到位判断阈值
M_SetStopDec	设置指定轴的停止加速度
M_GetStopDec	获取指定轴的停止加速度
M_GetSts	获取指定轴的轴状态
M_ClrSts	清除指定轴的限位与报警状态
M_GetCmd	获取指定轴的指令规划位置
M_GetCmdVel	获取指定轴的指令规划速度
M_GetEncPos	获取指定轴的编码器反馈位置
M_ReadActualPosition	获取指定轴的驱动器反馈位置
M_SetCurrentPos	设置指定轴的当前位置
M_GetTargetPos	获取指定轴的命令目标位置
M_GetEncVel	获取指定轴的编码器反馈速度
M_SetAxisStsSync	设置指定轴的限位信号锁定
M_GetAxisStsSync	获取指定轴的限位信号锁定
M_GetEcatDigitalInput	获取指定轴的伺服数字量输入数据
M_EcatStatusWord	获取指定轴的伺服 0X6041 映射
M_AxisPDOWrite	设置指定轴的 PDO 对象的数据
M_AxisPDORead	获取指定轴的 PDO 对象的数据

M_EcatSetOperationMode	设置指定轴的 6060 对象的值，即控制模式
M_EcatGetOperationMode	获取指定轴的 6061 对象的值，即控制模式

轴状态指令包含了轴的使能操作，轴参数的设置与获取，轴状态的获取等。使用轴状态指令可以获取到轴运动的信息，对于轴的监控十分有帮助。在使用运动指令之后可以通过 M_GetSts 来获取轴的运动状态来判断轴是否运动完成，以便进行下一步的运动。

例程

2.1 总线上共有 5 个轴，使能后获取每个轴的状态。

```
short ServoNum=5;
double[] ServoPos = new double[5];
double[] ServoVel = new double[5];
int[] ServoSts = new int[5];
double[] CmdPos = new double[5];
double[] CmdVel = new double[5];
for (short i = 1; i <= ServoNum; i++)
{
   ret = ecat_motion.M_Servo_On(i, CardID);
}
ret = ecat_motion.M_GetEncPos(1, out ServoPos [0], ServoNum, CardID);
//获取编码器位置
ret = ecat_motion.M_GetEncVel(1, out ServoVel[0], ServoNum, CardID);
//获取编码器速度
Ret = ecat_motion.M_GetSts(1, out ServoSts[0], ServoNum, CardID);
//获取轴运行状态
ret = ecat_motion. M_GetCmd (1, out CmdPos[0], .ServoNum, CardID);
//获取轴规划位置
ret = ecat_motion. M_GetCmdVel (1, out CmdVel[0], ServoNum, CardID);
//获取轴规划速度
```

2.2 获取轴1的运行状态，并且根据状态反应到UI的Label上

```
int Sts = ServoSts [0];
```

```
//伺服报警
if ((Sts & 0x02) == 0x02)
    Lab_Alarm.BackColor = Color.Red;
else
    Lab_Alarm.BackColor = SystemColors.ActiveCaptionText;
//原点光电信号
if ((Sts & 0x100000) == 0x100000)
    Lab_Org.BackColor = Color.Red;
else
    Lab_Org.BackColor = SystemColors.ActiveCaptionText;
//正向极限
if ((Sts & 0x20) == 0x20)
       Lab_Pel.BackColor = Color.Red;
else
       Lab_Pel.BackColor = SystemColors.ActiveCaptionText;
//反向极限
if ((Sts & 0x40) == 0x40)
    Lab_Mel.BackColor = Color.Red;
else
    Lab_Mel.BackColor = SystemColors.ActiveCaptionText;
       //伺服使能
     if ((Sts & 0x200) == 0x200)
           Lab_Servo.BackColor = Color.Red;
     else
          Lab_Servo.BackColor = SystemColors.ActiveCaptionText;
    //运动
     if ((Sts & 0x400) == 0x400)
          Lab_Motion.BackColor = Color.Red;
     else
          Lab_Motion.BackColor = SystemColors.ActiveCaptionText;
    //轴到位
     if ((Sts & 0x800) == 0x800)
          Lab_Inp.BackColor = Color.Red;
     else
          Lab_Inp.BackColor = SystemColors.ActiveCaptionText;
```

M_Servo_On

M_Servo_On	使能

指令说明: 使指定轴上使能。

指令原型： short M_Servo_On (short axis, short card)

参数说明：

axis	指定轴号，从 1 开始计数
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Servo_Off

M_Servo_Off	去使能

指令说明： 使指定轴去使能。

指令原型： short M_Servo_Off (short axis, short card)

参数说明：

axis	指定轴号，从 1 开始计数
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_SetSoftLimit

M_SetSoftLimit	设置指定轴的软件位置限制

指令说明： 设置指定轴的软件位置限制。

指令原型： short M_SetSoftLimit (short axis, int positive, int negative, short card)

参数说明：

axis	指定轴号，从 1 开始计数

positive	正向软件位置限制值
negative	负向软件位置限制值
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetSoftLimit

M_GetSoftLimit	获取指定轴的软件位置限制

指令说明: 获取指定轴的软件位置限制。

指令原型: short M_GetSoftLimit (short axis, out int pPositive,

out int pNegative, short card)

参数说明:

axis	指定轴号，从 1 开始计数
pPositive	正向软件位置限制值内存指针
pNegative	负向软件位置限制值内存指针
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_SetAxisBand

M_SetAxisBand	设置指定轴的到位判断阈值

指令说明: 设置指定轴的到位判断阈值，该函数可以设定轴的 INP 信号的判定条件，若不设置，默认为 0，则 INP 信号不会生效。

指令原型： short M_SetAxisBand (short axis, uint band, uint time, short card)

参数说明：

axis	指定轴号，从 1 开始计数
band	到位位置带设定，单位脉冲
time	到位稳定时间设定，单位 ms
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetAxisBand

M_GetAxisBand	获取指定轴的到位判断阈值

指令说明： 获取指定轴的到位判断阈值。

指令原型： short M_GetAxisBand (short axis, out uint pBand, out uint pTime, short card)

参数说明：

axis	指定轴号，从 1 开始计数
pBand	到位位置带内存指针
pTime	到位判断时间内存指针
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_SetStopDec

M_SetStopDec	设置指定轴的停止加速度

指令说明: 设置指定轴的停止加速度,针对 M_STOP 函数的调用的停止加速度。

指令原型: short M_SetStopDec (short axis, double dSmoothDec, double dEmergencyDec, short card)

参数说明:

axis	指定轴号，从 1 开始计数
dSmoothDec	平滑停止加速度
dEmergencyDec	急停停止加速度
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetStopDec

M_GetStopDec	获取指定轴的停止加速度

指令说明: 获取指定轴的停止加速度。

指令原型: short M_SetStopDec (short axis, out double dSmoothDec,out double dEmergencyDec, short card)

参数说明:

axis	指定轴号，从 1 开始计数
dSmoothDec	平滑停止加速度

dEmergencyDec	急停停止加速度
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetSts

M_GetSts	获取指定轴的轴状态

指令说明: 获取指定轴的轴状态，该函数可以一条获取多个轴的状态。

指令原型: short M_GetSts (short axis, out int pSts, short count, short card)

参数说明:

axis	指定起始轴号，从 1 开始计数
pSts	轴状态数组首地址，报警以及限位状态需要用 M_ClrSts 清除
count	需要读取状态的轴数量
card	卡片卡号，从 0 开始，默认按主板插槽号排序

轴状态说明:

Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
预留	预留	负向极限 (MEL)	正向极限 (PEL)	预留	预留	预留	伺服报警 (ALM)	预留
Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9
回零完成	回零错误	预留	预留	预留	预留	到位 (INP)	运动 (MOVE)	使能 (SVON)
....	....	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18
....	....	掉线 (OFFLINE)	预留	预留	预留	原点开关	预留	目标到达

返回值 ：详见第五章。

M_ClrSts

M_ClrSts	清除指定轴的限位与报警状态

指令说明: 获取指定轴的停止加速度。

指令原型： short M_ClrSts (short axis, short count, short card)

参数说明:

axis	指定起始轴号，从 1 开始计数
count	需要清除状态轴的数量
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetCmd

M_GetCmd	获取指定轴的指令规划位置

指令说明: 取指定轴的指令规划位置，该值相当于伺服驱动器内部位置（该值通过 M_ReadActualPosition 获取）+板卡的偏移值。若驱动器内部位置是 10000，板卡的偏移位置是-1000，那么 CMD 获取到的数值为 9000。

指令原型： short M_GetCmd (short axis, out double pValue, short count, short card)

参数说明:

axis	指定起始轴号，从 1 开始计数
pValue	轴规划位置数组首地址

count	需要读取规划位置的轴数量
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetCmdVel

M_GetCmdVel	获取指定轴的指令规划速度

指令说明: 获取指定轴的指令规划速度。

指令原型： short M_GetCmdVel (short axis, out double pValue, short count, short card)

参数说明:

axis	指定起始轴号，从 1 开始计数
pValue	轴规划位置数组首地址
count	需要读取规划速度的轴数量
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetEncPos

M_GetEncPos	获取指定轴的编码器反馈位置

指令说明: 获取指定轴的编码器反馈位置，该值相当于伺服驱动器内部位置（该值通过 M_ReadActualPosition 获取）+板卡的偏移值。若驱动器内部位置是

10000，板卡的偏移位置是-1000，那么 ENC 获取到的数值为 9000。

指令原型： short M_GetEncPos (short axis, out double pValue, short count, short card)

参数说明：

axis	指定起始轴号，从 1 开始计数
pValue	轴编码器反馈位置数组首地址
count	需要读取编码器反馈位置的轴数量
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_ReadActualPosition

M_ReadActualPosition	**获取指定轴的驱动器反馈位置**

指令说明： 获取指定轴的驱动器反馈位置。

指令原型： short M_ReadActualPosition (short axis, out int pPosition, short count, short card)

参数说明：

axis	指定起始轴号，从 1 开始计数
pPosition	轴驱动器反馈位置数组首地址
count	需要读取编码器反馈位置的轴数量
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_SetCurrentPos

M_SetCurrentPos	设置指定轴的当前位置

指令说明: 设置指定轴的当前位置，该函数会将伺服内部位置进行偏移计算，详情参见 M_GetCmd。

指令原型： short M_SetCurrentPos(short axis, int pos, short card = 0);

参数说明:

axis	指定起始轴号，从 1 开始计数
pos	设置当前位置值，设置以后轴的规划位置和反馈位置都会改变
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetTargetPos

M_GetTargetPos	获取指定轴的指令目标位置

指令说明: 获取指定轴的指令目标位置，该值相当于每条指令的终点位置。

指令原型： short M_GetTargetPos (short axis, ref int pValue, short count, short card)

参数说明:

axis	指定起始轴号，从 1 开始计数
pValue	轴命令目标位置数组首地址

count	需要读取的轴数量
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetEncVel

M_GetEncVel	获取指定轴的编码器反馈速度

指令说明: 取指定轴的编码器反馈速度。

指令原型： short M_GetEncPos (short axis, out double pValue, short count, short card)

参数说明:

axis	指定起始轴号，从 1 开始计数
pValue	轴编码器反馈速度数组首地址
count	需要读取编码器反馈速度的轴数量
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_SetAxisStsSync

M_ SetAxisStsSync	设置轴限位信号锁定

指令说明: 设置轴限位信号锁定，若锁定状态下，轴遮挡到限位以后再退出，限位信号将会保持。若干不锁定，轴遮挡限位以后再退出，限位信号将会自动清除。

指令原型： short M_ SetAxisStsSync (short axis, short on, short card = 0)

参数说明：

axis	指定起始轴号，从 1 开始计数
on	0：触发限位后锁定限位信号，需要用 ClrSts 清除 1：触发限位后再退出限位，状态自动清除
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetAxisStsSync

M_ GetAxisStsSync	获取轴限位信号锁定

指令说明： 获取轴限位信号锁定。

指令原型： short M_GetAxisStsSync (short axis, short* on, short card = 0)

参数说明：

axis	指定起始轴号，从 1 开始计数
on	0：触发限位后锁定限位信号，需要用 ClrSts 清除 1：触发限位后再退出限位，状态自动清除
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetEcatDigitalInput

M_GetEcatDigitalInput	获取指定轴的伺服数字量输入数据

指令说明: 获取指定轴的伺服数字量输入数据，获取 0x60fd 的数据。

指令原型： short M_GetEcatDigitalInput (short axis, out uint digitalInput, short card)

参数说明:

axis	指定轴号，从 1 开始计数
digitalInput	对应伺服的数字量输入信号（60FD），可获取原点限位信号，Bit2 为原点信号，限位推荐使用 M_GetSts 获取
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_EcatStatusWord

M_EcatStatusWord	获取指定轴的伺服 0X6041 映射

指令说明: 获取指定轴的伺服 0X6041 映射。

指令原型： short M_EcatStatusWord (short axis, ref ushort statusword, short card)

参数说明:

axis	指定轴号，从 1 开始计数
statusword	对应伺服的状态字（6041），具体用法请参考伺服的

	Ethercat 通信手册。
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_AxisPDOWrite

M_AxisPDOWrite	**设置指定轴的 PDO 对象的数据**

指令说明: 设置指定轴的 PDO 对象的数据。

指令原型: short M_AxisPDOWrite(short axis, short index, short subindex, uint data, short data_size, short card = 0)

参数说明:

axis	指定轴号，从 1 开始计数
index	PDO 映射地址,只有 eni 中包含的 TXPDO 才可以设置。
subindex	PDO 映射地址的子索引。
data	需要设置的 PDO 数据
data_size	数组长度，单位 byte
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_AxisPDORead

M_AxisPDORead	**获取指定轴的 PDO 对象的数据**

指令说明: 获取指定轴的 PDO 对象的数据。

指令原型: short M_AxisPDORead(short axis, short index, short subindex, short data_size, ref uint pBuf, short count = 0, short card = 0)

参数说明:

axis	指定轴号，从 1 开始计数
index	PDO 映射地址,只有 eni 中包含的 RXPDO 才可以读取。
subindex	PDO 映射地址的子索引。
data_size	数组长度，单位 byte
pBuf	需要读取的数据数组首地址
Count	需要读取的 pdo 数量
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_EcatSetOperationMode

M_EcatSetOperationMode	**设置指定轴的 6060 对象的值，即控制模式**

指令说明: 设置指定轴的 6060 对象的值，即控制模式。

指令原型: short M_EcatSetOperationMode(short axis, short mode, short card)

参数说明:

axis	指定轴号，从 1 开始计数
Mode	控制模式，标准 Ethercat 协议规定的伺服控制模式。

	6-回零 8-CSP(同步位置，板卡控制位置模式) 9-CSV(同步速度) 10-CST(同步扭矩)
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_EcatGetOperationMode

M_EcatGetOperationMode	**获取指定轴的 6061 对象的值，即控制模式**

指令说明: 获取指定轴的 6061 对象的值，即控制模式。

指令原型： short M_EcatGetOperationMode(short axis, ref short mode, short card)

参数说明:

axis	指定轴号，从 1 开始计数
Mode	控制模式，标准 Ethercat 协议规定的伺服控制模式。 6-回零 8-CSP(同步位置，板卡控制位置模式) 9-CSV(同步速度) 10-CST(同步扭矩)
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

三.运动指令

1.绝对相对运动

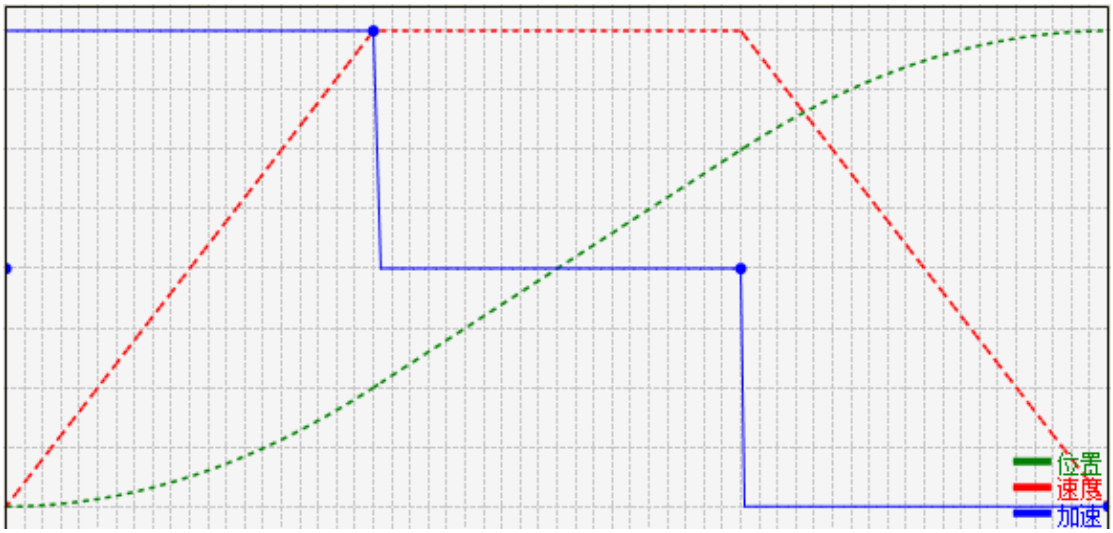
指令	说明
M_SetMove	设定指定轴的绝对和相对运动加速度
M_GetMove	获取指定轴的绝对和相对运动加速度
M_AbsMove	指定轴开始绝对运动
M_RelMove	指定轴开始相对运动

绝对运动的参考原点是系统回零后的原点，通过 M_SetMove 设置运动的加减速度以及加加速时间，然后用 M_AbsMove 运行绝对运动，速度和目标位置在该条函数中设置。

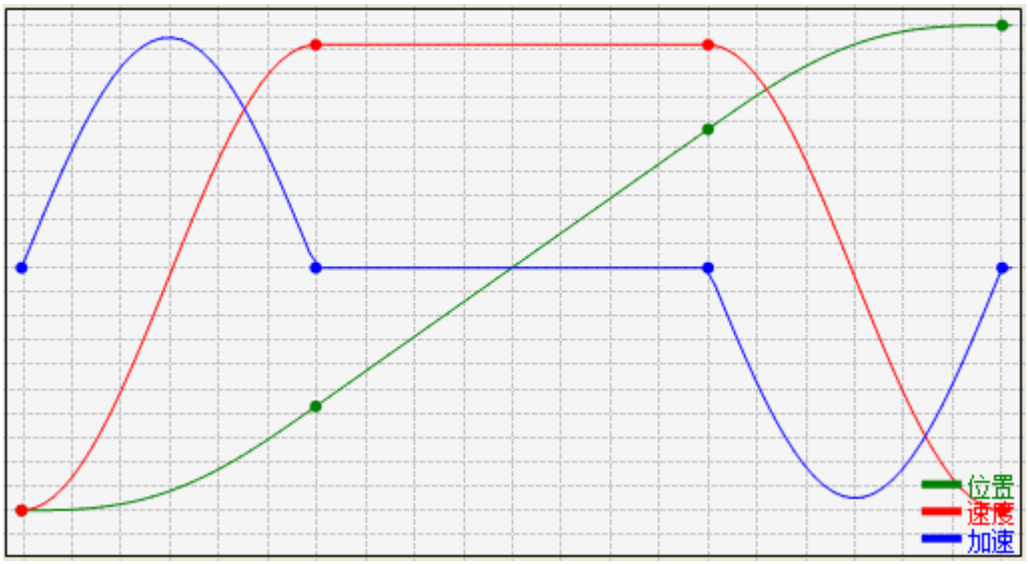
相对运动是根据当前位置移动设定的距离，通过 M_SetMove 设置运动的加减速度以及加加速时间，然后用 M_RelMove 运行相对运动，速度和运动距离在该条函数中设置。

加加速时间含义:在一段完整的点对点运动中，轴会经历加速，匀速，减速的

不同阶段，同时产生三种不同的时间曲线。如图所示，绿色是时间位移曲线，红色是时间速度曲线，蓝色是时间加速度曲线。曲线由 2 个点被分割成 3 段，第一段加速，第二段匀速，第三段减速。



该图所示的蓝色曲线为加速度时间曲线，可以看到加速度是突变的，两个值之间的切换用时几乎忽略。而速度曲线也是对应的 T 型曲线。 这样的运动会使机构在加速和减速的时候会有一定的震动。



可以看到上图的加加速度加入了一个加速度变化的时间，使之并不是突变的。从而使速度曲线在变化的过程中更加平滑，也就是我们常说的 S 型曲线加速。对于机械的震动起到一定的缓解作用。Stime 描述的就是整个运动过程中的加速度的变化时间，设定的范围是 0~200ms，时间越长，加速度的变化时间越长，加速越平滑。

例程

3.1.1 命令轴 1 以 100000 脉冲/s 的速度，1000000 脉冲/s/s 的加速度，100ms 的加加速时间运动到 250000 脉冲的位置。

```
ecat_motion.CmdPrm CmdPrm = new ecat_motion.CmdPrm();
//点位运动参数
int ACC = 1000000, DEC = 1000000,stime=100, POS = (int)250000, VEL =
100000;
//设置参数数值
CmdPrm.acc = ACC;
CmdPrm.dec = DEC;
CmdPrm.stime= stime;
int ret = ecat_motion. M_SetMove (1, ref CmdPrm, CardID);
//将参数写入板卡
ecat_motion. M_AbsMove (1, POS, VEL, CardID); //开始绝对运动
```

3.1.2 命令轴 1 以 100000 脉冲/s 的速度，1000000 脉冲/s/s 的加速度，100ms 的加加速时间，运动 250000 脉冲的距离。

```
ecat_motion.CmdPrm CmdPrm = new ecat_motion.CmdPrm();
//点位运动参数
int ACC = 1000000, DEC = 1000000, stime=100, DIS = (int)250000, VEL =
100000;
//设置参数数值
CmdPrm.acc = ACC;
CmdPrm.dec = DEC;
CmdPrm.stime=  stime;
//将参数写入板卡
```

```
int ret = ecat_motion.M_SetMove (1, ref CmdPrm, CardID);
ecat_motion.M_RelMove (1, DIS, VEL, CardID); //开始相对运动
```

M_SetMove

M_SetMove	设定指定轴的绝对和相对运动加速度

指令说明: 设置指定轴的绝对和相对运动的加速度。

指令原型: short M_SetMove (short axis, ref CmdPrm pPrm, short card)

参数说明:

axis	指定轴号，从 1 开始计数
pPrm	加速度结构体 CmdPrm 变量名称
card	卡片卡号，从 0 开始，默认按主板插槽号排序

CmdPrm:

变量名	Acc	Dec	Stime
说明	加速度	减速度	加加速时间

返回值 ：详见第五章。

M_GetMove

M_GetMove	获取指定轴的绝对和相对运动加速度

指令说明: 获取指定轴的绝对和相对运动的加速度。

指令原型: short M_GetMove (short axis, out CmdPrm pPrm, short card)

参数说明:

axis	指定轴号，从 1 开始计数

pPrm	加速度结构体 CmdPrm 变量名称
card	卡片卡号，从 0 开始，默认按主板插槽号排序

CmdPrm:

变量名	Acc	Dec	Stime
说明	加速度	减速度	加加速时间

返回值 ：详见第五章。

M_AbsMove

M_AbsMove	**指定轴开始绝对运动**

指令说明: 指定轴开始绝对运动。

指令原型: short M_AbsMove (short axis, int pos, double vel, short card)

参数说明:

axis	指定轴号，从 1 开始计数
pos	绝对运动的目标位置
vel	绝对运动的指令目标速度
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_RelMove

M_RelMove	**指定轴开始相对运动**

指令说明: 指定轴开始相对运动。

指令原型： short M_RelMove (short axis, int pos, double vel, short card)

参数说明：

axis	指定轴号，从 1 开始计数
pos	相对运动的运动距离
vel	相对运动的指令目标速度
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

2.Jog 运动

指令	说明
M_Jog	指定轴开始 Jog 运动

Jog 运动是根据通过函数进行对轴的位置微动操作，通过 M_SetMove 设置运动的加减速度，然后用 M_Jog 运行 Jog 运动，速度在该条函数中设置，使用 M_Stop 停止运动。

例程

3.2.1 命令轴 1 以 1000 脉冲/s 的速度 JOG 运动 5s 后停止。

```
int AxisNum=1;
```

```
ecat_motion.CmdPrm TarpPrfPrm = new ecat_motion.CmdPrm();
//JOG运动参数
int ACC = 1000000, DEC = (int)1000000,VEL=1000; //JOG运动参数设置
CmdPrm.acc = ACC;CmdPrm.dec = DEC;
int ret= ecat_motion. M_SetMove (AxisNum, ref CmdPrm, CardID);
//参数设置到卡片
ret = ecat_motion.M_Jog(AxisNum, VELL, CardID); //开始 JOG 运动
System.Threading.Thread.Sleep(5000);
ret =ecat_motion.M_Stop((int)(1 << (short)(AxisNum - 1)), 0, CardID);
//以平滑方式停止 jog
```

M_Jog

M_Jog	指定轴开始 Jog 运动

指令说明: 指定轴开始 Jog 运动。

指令原型: short M_Jog (short axis, double vel, short card)

参数说明:

axis	指定轴号，从 1 开始计数
vel	Jog 运动的指令目标速度
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

3.停止运动

指令	说明
M_Stop	指定轴停止当前的运动
M_StopSingleAxis	指定单轴停止当前的运动

停止运动指令能够使指定轴立即停止当前正在执行的运动，包括绝对运动，相对运动以及 Jog 运动，不包含回零运动。停止分为两种模式，一种是缓停，一种是急停。两种停止的减速度通过 M_SetStopDec 来设置。

M_Stop

M_Stop	指定轴停止当前的运动

指令说明: 指定轴停止当前的运动。

指令原型： short M_Stop (uint mask, short option, short card)

参数说明:

mask	按位指示需要对应停止的轴号，Bit0 为轴 1，Bit1 为轴 2，以此类推
option	停止运动的方式，0 为平滑减速度停止，1 为急停减速度停止，参数设置详见 M_SetStopDec
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_StopSingleAxis

M_StopSingleAxis	指定轴停止当前的运动

指令说明: 指定轴停止当前的运动。

指令原型： short M_StopSingleAxis (short axis, int option, short card)

参数说明:

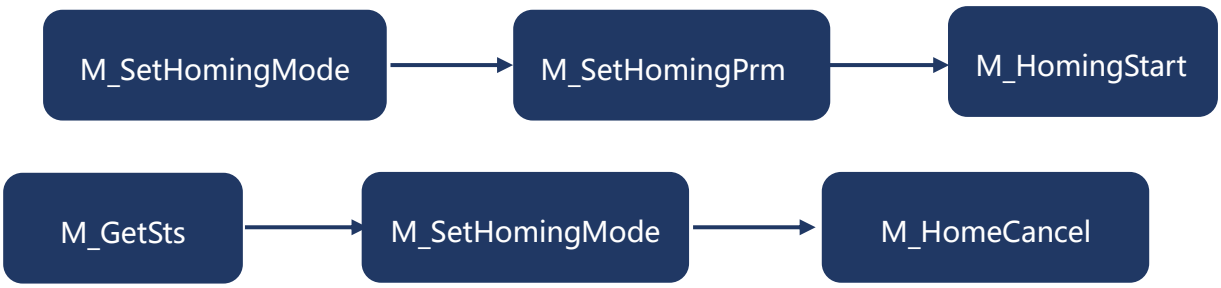
axis	轴号，从 1 开始计数
option	停止运动的方式，0 为平滑减速度停止，1 为急停减速度停止，参数设置详见 M_SetStopDec
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

4.回零运动

指令	说明
M_SetHomingMode	设置指定轴的运动模式
M_SetHomingPrm	设置指定轴的回零参数
M_GetHomingPrm	获取指定轴的回零参数
M_HomingStart	指定轴开始回零运动
M_WaitHomingFinished	等待指定轴回零完成
M_GetEcatHomingStatus	获取指定轴的回零状态
M_HomeCancel	指定轴停止当前的回零运动
M_HomeCancelSingleAxis	指定单轴停止当前的回零运动

回零运动是直接使用驱动器的回零方法。驱动器的回零模式需要参考驱动器的 Ethercat 指令手册。回零前需要用 M_SetHomingMode 将驱动器的模式切换到 6，即回零模式，然后再运行开始回零，结束以后需要将驱动器的模式切换到 8,并且执行 M_HomeCancel。注意：回零速度单位请查看驱动器说明书！



例程

3.4.1 命令轴 1 以驱动器模式 33 回零（一般是负向寻 EZ，具体请查阅驱动器手册）。

```
int ret = ecat_motion.M_SetHomingMode(AxisNum, 6,CardID);
//使用驱动器的回零模式，一般为 6，CSP 为 8
int OFFSET = 0; uint SPEED1 = 1000, SPEED2 = 2000, ACC = 10000;
//设置回零参数
ecat_motion.M_ SetHomingPrm (AxisNum, 33, OFFSET, SPEED1, SPEED2, ACC,
0, CardID);

//开始回零运动
ecat_motion.M_ HomingStart(AxisNum, CardID);
ushort Sts = 0;
do
{
   System.Threading.Thread.Sleep(10);
   ret = ecat_motion.M_GetSts(i, ref Sts, 1 ,CardID);
   //获取6041状态字
}
while ((Sts & 0x400) == 0x400);
//判断轴的MOVE信号
ret = lctdevice.ecat_motion.M_SetHomingMode(AxisNum, 8, global.CardID);
//切换到驱动器 CSP 模式，一般为 6，CSP 为 8
ret = lctdevice.ecat_motion.M_HomeCancel((ushort)(1 << (short)(AxisNum-
1)),CardID);
//回零完成
```

M_SetHomingMode

M_SetHomingMode	设置指定轴的运动模式

指令说明: 设置指定轴的运动模式。

指令原型: short M_SetHomingMode (short axis, short mode, short card)

参数说明:

axis	指定轴号，从 1 开始计数
mode	基于 CIA402 协议规定的运动模式，模式 6 是驱动器回零模式，模式 8 是 CSP 模式即板卡控制模式
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_SetHomingPrm

M_SetHomingPrm	设置指定轴的驱动器回零参数

指令说明: 设置指定轴的驱动器回零参数。

指令原型: short M_SetHomingPrm (short axis, short method, int offset, uint speed1, uint speed2, uint acc, ushort probeFunction, short card)

参数说明:

axis	指定轴号，从 1 开始计数
method	驱动器回零方法，参见驱动器的说明书中的回零部分
offset	回零完成后的偏移距离

speed1	第一回零速度，详情参见驱动器的说明书中的回零部分
speed2	第二回零速度，详情参见驱动器的说明书中的回零部分
acc	回零加速度，详情参见驱动器的说明书中的回零部分
probeFunction	预留，默认为 0
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetHomingPrm

M_GetHomingPrm	获取指定轴的驱动器回零参数

指令说明: 获取指定轴的驱动器回零参数。

指令原型: short M_GetHomingPrm (short axis,ref short method, ref int offset, ref uint speed1, ref uint speed2, ref uint acc, ref ushort probeFunction, short card)

参数说明:

axis	指定轴号，从 1 开始计数
method	驱动器回零方法，参见驱动器的说明书中的回零部分
offset	回零完成后的偏移距离
speed1	第一回零速度，详情参见驱动器的说明书中的回零部分
speed2	第二回零速度，详情参见驱动器的说明书中的回零部分
acc	回零加速度，详情参见驱动器的说明书中的回零部分
probeFunction	预留，默认为 0

card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_HomingStart

M_HomingStart	指定轴开始进行驱动器回零

指令说明: 指定轴开始进行驱动器回零。

指令原型： short M_HomingStart (short axis，short card)

参数说明:

axis	指定轴号，从 1 开始计数
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_WaitHomingFinished

M_WaitHomingFinished	等待回零完成

指令说明: 等待轴回零完成，该函数为阻塞型函数。

指令原型： short M_WaitHomingFinished (short axis，short card)

参数说明:

axis	指定轴号，从 1 开始计数
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetEcatHomingStatus

M_GetEcatHomingStatus	获取指定轴的回零状态

指令说明: 获取指定轴的回零状态。

指 令 原 型 : short M_GetEcatHomingStatus (short axis, out short phomingStatus, short card)

参数说明:

axis	指定轴号，从 1 开始计数
phomingStatus	指定轴的回零状态返回值，根据 CIA402 标准协议的规定状态说明定义，不同驱动器可能会有出入
card	卡片卡号，从 0 开始，默认按主板插槽号排序

回零状态表:

返回值	-1	1	2	3	4
状态	回零错误	正在回零	未获取到	回零成功	回零中断

返回值 ：详见第五章。

M_HomeCancel

M_HomeCancel	指定轴停止当前的回零运动

指令说明: 指定轴停止当前的回零运动。

指令原型: short M_HomeCancel (uint mask, short card)

参数说明:

mask	按位指示需要对应停止回零的轴号，Bit0 为轴 1，Bit1

	为轴 2，以此类推
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_HomeCancelSingleAxis

M_HomeCancelSingleAxis	指定单轴停止当前的回零运动

指令说明: 指定单轴停止当前的回零运动。

指令原型： short M_HomeCancelSingleAxis (short axis, short card)

参数说明:

axis	轴号，从 1 开始计数
card	卡片卡号，从 0 开始，默认按主板插槽号排序

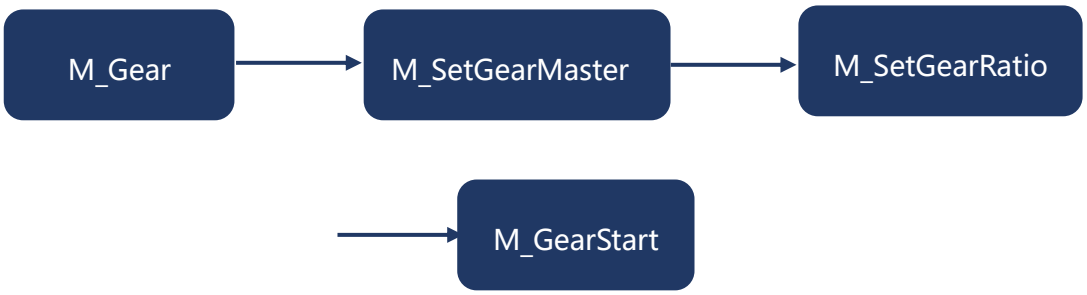
返回值 ：详见第五章。

5.电子齿轮运动

指令	说明
M_Gear	将从指定轴切换到电子齿轮模式
M_SetGearMaster	设置主轴以及从轴的跟随对象
M_SetGearRatio	设置电子齿轮的齿轮比
M_GearStart	从轴开启电子齿轮运动
M_GearStartSingleAxis	指定单轴开始执行电子齿轮运动

M_GetGearMaster	获取电子齿轮的主轴以及跟随对象
M_GetGearRatio	获取电子齿轮的齿轮比

电子齿轮运动是规定从轴跟随主轴的一种控制方式，跟随按照电子齿轮比的关系进行绑定，同一个主轴可以有多个从轴按照不同的电子齿轮比进行跟随。常见的应用场景是龙门模式，从轴跟随主轴按 1:1 的齿轮比进行绑定。此时控制主轴，从轴会跟随运动。



M_Gear

M_Gear	将从指定轴切换到电子齿轮模式

指令说明: 将从指定轴切换到电子齿轮模式。

指令原型： short M_Gear (short axis, short dir, short card)

参数说明:

axis	轴号，从 1 开始计数
dir	跟随方向，0：双向跟随，1：正向跟随，-1：负向跟随
card	卡号

返回值 ：详见第五章。

M_SetGearMaster

M_SetGearMaster	设置主轴以及从轴的跟随对象

指令说明: 设置主轴以及从轴的跟随对象。

指令原型: short M_SetGearMaster(short axis, short masterindex, short masterType, short masterItem, short card)

参数说明:

axis	轴号，从 1 开始计数
masterindex	主轴轴号，从 1 开始计数
masterType	主轴对象，2: 主轴规划位置 1：主轴编码器反馈 3：主轴 4：主轴规划输出值 5：主轴编码器输出值
masterItem	轴类型，masterType 为 3 时生效 0 表示主轴规划输出值 1 表示主轴的编码器输出
card	卡号

返回值 ：详见第五章。

M_SetGearRatio

M_SetGearRatio	设置电子齿轮比

指令说明: 设置电子齿轮比。

指令原型： short M_SetGearRatio(short axis, int masterEven, int slaveEven, int masterSlope, short card);

参数说明：

axis	轴号，从 1 开始计数
masterEven	主轴的传动比系数
slaveEven	从轴的传动比系数
masterSlope	主轴离合区位移，范围>=0 并且!=1
card	卡号

返回值 ：详见第五章。

M_GearStart

M_GearStart	设置电子齿轮比

指令说明： 设置电子齿轮比。

指令原型： short M_GearStart (uint mask, short card);

参数说明：

mask	按位指示的轴号，Bit0 为轴 1，Bit1 为轴 2，以此类推
card	卡号

返回值 ：详见第五章。

M_GearStartSingleAxis

M_GearStartSingleAxis	指定单轴开始电子齿轮运动

指令说明: 指定单轴开始电子齿轮运动。

指令原型: short M_GearStartSingleAxis (short axis, short card);

参数说明:

axis	轴号，从 1 开始计数
card	卡号

返回值 ：详见第五章。

M_GetGearMaster

M_ GetGearMaster	获取电子齿轮主轴以及跟随对象

指令说明: 获取电子齿轮的主轴以及跟随对象。

指令原型: short M_ GetGearMaster (short axis, out short masterindex, out short masterType, out short masterItem, short card);

参数说明:

axis	轴号，从 1 开始计数
masterindex	主轴轴号，从 1 开始计数
masterType	主轴对象，2: 主轴规划位置 1：主轴编码器反馈 3：主轴 4：主轴规划输出值 5：主轴编码器输出值
masterItem	轴类型，masterType 为 3 时生效

	0 表示主轴规划输出值 1 表示主轴的编码器输出
card	卡号

返回值 ：详见第五章。

M_GetGearRatio

M_GetGearRatio	获取电子齿轮比

指令说明: 设置电子齿轮比。

指令原型: short M_GetGearRatio(short axis,out int masterEven,out int slaveEven,out int masterSlope, short card);

参数说明:

axis	轴号，从 1 开始计数
masterEven	主轴的传动比系数
slaveEven	从轴的传动比系数
masterSlope	主轴离合区位移，范围>=0 并且!=1
card	卡号

返回值 ：详见第五章。

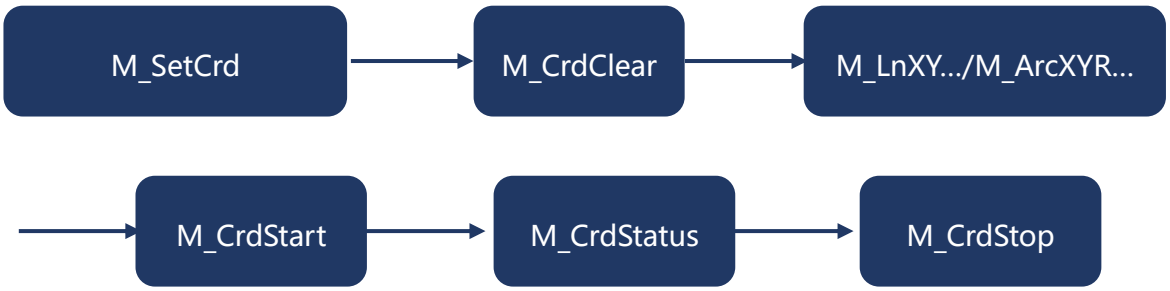
6.连续插补运动

指令	说明
M_SetCrd	设置连续插补的坐标系参数
M_GetCrd	获取连续插补的坐标系参数
M_CrdClear	清除坐标系与 FIFO 中的位置缓存
M_Line	向缓存区压入直线插补命令
M_Arc2R	半径基准向缓存区压入 2 维圆弧插补命令
M_Arc2C	圆心基准向缓存区压入 2 维圆弧插补命令
M_BufIO	向缓冲区压入 DO 命令
M_BufDelay	向缓冲区压入延时命令
M_GetLastCrdPos	获取缓冲区中最后一个点的坐标
M_CrdStatus	获取缓冲区运行状态
M_CrdStart	使对应坐标系开始运行连续插补运动
M_CrdStop	使对应坐标系停止连续插补运动
M_CrdPause	使指定坐标系暂停当前的动作
M_CrdContinue	使指定坐标系继续当前的动作

连续插补运动是使设定的轴按照规划的轨迹进行运动的一种模式，客户只需要将点位提前压入 FIFO 中，执行开始运动以后运动会自动进行。通过监控状态来判断轴运动完成。运动包含了 2-4 轴直线插补，2 轴圆弧插补，DO 点位输出以及延时等待命令。

PCIe-M60 共有两套连续插补坐标系,每套坐标系共有 2 个 FIFO,每个 FIFO 可以存放 512 个点。每个坐标系之间相互独立。直线插补点，圆弧插补点，DO 点位输出以及延时等待命令都会占用一个点的空间。

通过连续插补指令，客户只需要将路径设置好，板卡会自动处理规划轴的行动轨迹以及轨迹速度。每一段轨迹都可以设定不同的速度和加速度，使轨迹的设置更加灵活。



例程

3.6.1 建立坐标系，初始化 FIFO

```
ecat_motion.CrdCfg crdCfg = new ecat_motion.CrdCfg();
//连续插补参数设置
crdCfg.axis = new short[8]; //插补轴数组
crdCfg.orignPos = new int[8]; //插补起始坐标数组
crdCfg.dimension = 2;//插补维度
crdCfg.axis[0] = x;//第一个轴
crdCfg.axis[1] = y;//第二个轴
crdCfg.axis[2] = z;//第三个轴
crdCfg.axis[3] = r;//第四个轴
for (int i = 0; i < crdCfg.dimension; i++)
{
    crdCfg.orignPos[i] = 0;//起点坐标清零
}
//设置插补参数
crdCfg.evenTime = 10; //最小匀速时间 单位 cycletime（默认1ms）
crdCfg.synVelMax =  1000000;//插补速度 pulse/S
crdCfg.synAccMax =  100000;//插补加速度 pulse/S/S
crdCfg.synDecSmooth = 1000000; //插补平滑减速度 pulse/S/S
```

```
crdCfg.synDecAbrupt = 5000000; //插补急停减速度 pulse/S/S
int ret = ecat_motion.M_SetCrd (1, ref crdCfg, CardID);
//设置插补坐标系参数到板卡
ret = ecat_motion.M_CrdClear(1, 1, 0, CardID); //清除FIFO缓存
```

3.6.2 向 FIFO 中插入点位后开始运动，并且监控插补状态，判断完成

```
int vel=10000; //单段插补速度，单位pulse/S
int acc=100000; //单端插补加速度,单位pulse/S/S
int sSpace = 0;//Buffer已用空间
short sSts = 0;//Buffer状态
short sCmdNum = 0;//Buffer已有命令数
int ret = ecat_motion.M_Line(......); //插入直线
ret = ecat_motion.M_Line(......); //插入直线
ret = ecat_motion.M_CrdStart(1, 0, CardID); //开始插补运动
do
{
ecat_motion.M_CrdStatus(1,out sSts,out sCmdNum,out sSpace,0, CardID);
    //获取FIFO状态
      System.Threading.Thread.Sleep(100);
}
while (sSts == 1 && sCmdNum != 0); //判断FIFO状态以及命令数量
//插补完成
ecat_motion.M_CrdStop(1, 0, CardID); //关闭插补功能
```

M_SetCrd

M_SetCrd	设置连续插补的坐标系参数

指令说明: 设置连续插补的坐标系参数，可以设定坐标系的起点坐标、插补维度以及插补的最大速度和加速度等参数。

指令原型: short M_SetCrd (short crd, ref CrdCfg pCrdPrm, short card)

参数说明:

crd	坐标系编号，可输入 1,2

pCrdPrm	坐标系参数结构体 CrdPrm 的内存指针
card	卡片卡号，从 0 开始，默认按主板插槽号排序

结构体 PCrdPrm：

short dimension	维度，设定坐标系的维度，最高可设置 4 维 XYZA
short axis [8]	坐标系每个轴对应的实际轴号，对应关系为 XYZA
short setOriginFlag	设置原点坐标值标志，0:默认当前规划位置为原点位置;1:用户指定原点位置
long originPos [8]	用户指定的原点位置，分别对应 XYZA4 轴的坐标
short evenTime	最小匀速时间，使每段插补能够强制匀速运动的最小时间，若速度曲线只有加速度段和减速段，可以在尖峰处形成匀速的水平直线，减小速度突变带来的冲击，单位 cycletime（默认 1ms）
double synVelMax	坐标系的最大限制插补速度，分段速度大于该速度会默认按该速度运动，单位 pulse/S
double synAccMax	坐标系的最大限制插补加速度，分段加速度大于该加速度会默认按该加速度运动，单位 pulse/S/S
double synDecSmooth	平滑停止减速度，单位 pulse/S/S
double synDecAbrupt	紧急停止减速度，单位 pulse/S/S

返回值 ：详见第五章。

M_GetCrd

M_GetCrd	获取连续插补的坐标系参数

指令说明: 获取连续插补的坐标系参数。

指令原型: short M_GetCrd (short crd, ref CrdCfg pCrdPrm, short card)

参数说明:

crd	坐标系编号，可输入 1,2
pCrdPrm	坐标系参数结构体 CrdPrm 的内存指针
card	卡片卡号，从 0 开始，默认按主板插槽号排序

结构体 PCrdPrm:

short dimension	维度，设定坐标系的维度，最高可设置 4 维 XYZA
short axis [8]	坐标系每个轴对应的实际轴号，对应关系为 XYZA
short setOriginFlag	设置原点坐标值标志，0:默认当前规划位置为原点位置;1:用户指定原点位置
long originPos [8]	用户指定的原点位置，分别对应 XYZA4 轴的坐标
short evenTime	最小匀速时间，使每段插补能够强制匀速运动的最小时间，若速度曲线只有加速度段和减速段，可以在尖峰处形成匀速的水平直线，减小速度突变带来的冲击，单位 cycletime（默认 1ms）
double synVelMax	坐标系的最大限制插补速度，分段速度大于该速度会默认按该速度运动，单位 pulse/S
double synAccMax	坐标系的最大限制插补加速度，分段加速度大于该

	加速度会默认按该加速度运动，单位 pulse/S/S
double synDecSmooth	平滑停止减速度，单位 pulse/S/S
double synDecAbrupt	紧急停止减速度，单位 pulse/S/S

返回值 ：详见第五章。

M_CrdClear

M_CrdClear	清除坐标系与 FIFO 中的位置缓存

指令说明: 清除坐标系与 FIFO 中的位置缓存

指令原型: short M_CrdClear (short crd, short count, short fifo , short card)

参数说明:

crd	坐标系编号，可输入 1,2
count	计数，默认为 1
fifo	先入先出缓存器，从 0 开始计数，共两个，0 和 1
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Line

M_Line	向缓存区压入直线插补命令

指令说明: 向缓存区压入直线插补命令，最多 512 个点

指令原型：short M_Line(short crd, short dimension, ref short axisArray, ref int posArray, double mVel, double acc, double velEnd = 0, short fifo = 0, short card = 0);

参数说明：

crd	坐标系编号，可输入 1,2
dimension	插补维度
axisArray	插补的参与轴数组首地址
posArray	插补位置数组首地址，位置元素和轴数组中的元素一一对应
mVel	插补的最大线速度
acc	插补的最大加速度
velEnd	插补的结束速度
fifo	先入先出缓存器，从 0 开始计数，共两个，0 和 1
card	卡号

返回值 ：详见第五章。

M_Arc2R

M_Arc2R	半径基准向缓存区压入 2 维圆弧插补命令

指令说明：向缓存区压入 2 维圆弧插补命令，最多 512 个点

指令原型： short M_Arc2R (short crd, short axisArray, short posArray, double radius, short circleDir, double mvel, double synAcc, short fifo,

short card)

参数说明:

crd	坐标系编号，可输入 1,2
axisArray	参与插补的轴数组首地址
posArray	圆弧的终点位置数组首地址
radius	圆弧半径
circleDir	旋转方向 0：顺时针 1：逆时针
synVel	该段运动的最大速度，若大于坐标系速度以坐标系速度运行
synAcc	该段运动的最大加速度，若大于坐标系加速度以坐标系加速度运行
fifo	先入先出缓存器，从 0 开始计数，共两个，0 和 1
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Arc2C

M_Arc2C	圆心基准向缓存区压入 2 维圆弧插补命令

指令说明: 向缓存区压入 2 维圆弧插补命令，最多 512 个点

指令原型: short M_Arc2C (short crd, short axisArray, short posArray, double centerArray, short circleDir, double mvel, double synAcc, short fifo, short card)

参数说明：

crd	坐标系编号，可输入 1,2
axisArray	参与插补的轴数组首地址
posArray	圆弧的终点位置数组首地址
centerArray	圆弧的圆心数组首地址，圆心坐标为相对坐标
circleDir	旋转方向 0：顺时针 1：逆时针
synVel	该段运动的最大速度，若大于坐标系速度以坐标系速度运行
synAcc	该段运动的最大加速度，若大于坐标系加速度以坐标系加速度运行
fifo	先入先出缓存器，从 0 开始计数，共两个，0 和 1
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_BufIO

M_BufIO	向缓冲区压入 DO 命令

指令说明： 向缓冲区压入 DO 命令

指令原型： short M_BufIO (short crd, short channel, short value, short fifo, short card)

参数说明：

crd	坐标系编号，可输入 1,2

channel	DO 的通道号
value	设置 DO 输出的状态 0：无输出，1：有输出
fifo	先入先出缓存器，从 0 开始计数，共两个，0 和 1
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_BufDelay

M_BufDelay	向缓冲区压入延时命令

指令说明: 向缓冲区压入延时命令

指令原型: short M_BufDelay (short crd, unsigned long delayTime，short fifo, short card)

参数说明:

crd	坐标系编号，可输入 1,2
delayTime	延迟的时间，单位是总线周期。范围是 1~65535
fifo	先入先出缓存器，从 0 开始计数，共两个，0 和 1
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_GetLastCrdPos

M_GetLastCrdPos	获取缓冲区中最后一个点的坐标

指令说明: 获取缓冲区中最后一个点的坐标

指令原型： short M_GetLastCrdPos (short crd, out int position,short fifo, short card)

参数说明:

crd	坐标系编号，可输入 1,2
position	传出的数组首地址
fifo	先入先出缓存器，从 0 开始计数，共两个，0 和 1
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_CrdStatus

M_CrdStatus	获取缓冲区运行状态

指令说明: 获取缓冲区的运行状态

指令原型： short M_CrdStatus (short crd, out short pSts, out short pCmdNum, out int pSpace, short fifo, short card)

参数说明:

crd	坐标系编号，可输入 1,2
pSts	插补运动状态 0：不在运动，1：正在运动
pCmdNum	FIFO 已存的命令空间
pSpace	FIFO 剩余空间
fifo	先入先出缓存器，从 0 开始计数，共两个，0 和 1

card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_CrdStart

M_CrdStart	使对应坐标系开始运行连续插补运动

指令说明: 使对应坐标系开始运行连续插补运动

指令原型： short M_CrdStart (short mask, short option, short card)

参数说明:

mask	mask 的 bit0 为 1：启动坐标系 1，bit1：坐标系 2
option	option 的 bit0 为坐标系 1，bit1 为坐标系 2。对应位为 0 启用 FIFO0，为 1 启用 FIFO1
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_CrdStop

M_CrdStop	使对应坐标系开始停止连续插补运动

指令说明: 使对应坐标系开始停止连续插补运动

指令原型： short M_CrdStop (short mask, short option, short card)

参数说明:

mask	mask 的 bit0 为 1：启动坐标系 1，bit1：坐标系 2

option	Option 是对应的停止方法，0：平滑停止，1：急停
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_CrdPause

M_CrdPause	使对应坐标系暂停连续插补运动

指令说明: 使对应坐标系暂停连续插补运动。使用暂停和继续函数时，只针对每个 CRD 的 FIFO-0 生效。FIFO-1 无效。

指令原型: short M_CrdPause (short crd, short card)

参数说明:

crd	坐标系号，1 或者 2
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_CrdContinue

M_CrdContinue	使对应坐标系继续连续插补运动

指令说明: 使对应坐标系继续连续插补运动。使用暂停和继续函数时，只针对每个 CRD 的 FIFO-0 生效。FIFO-1 无效。

指令原型: short M_CrdContinue (short crd, short card)

参数说明:

crd	坐标系号，1 或者 2
card	卡片卡号，从 0 开始，默认按主板插槽号排序

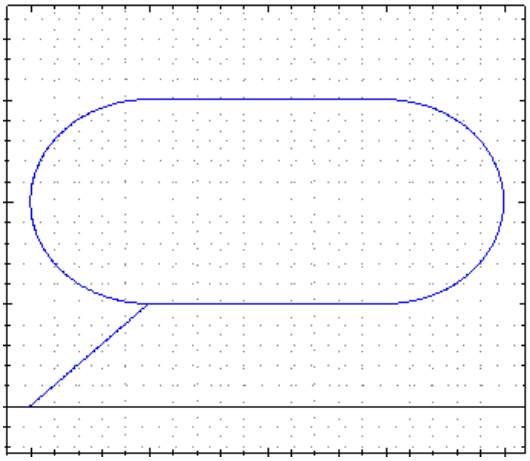
返回值 ：详见第五章。

7.速度前瞻功能

指令	说明
M_SetVelPlanning	初始化速度前瞻功能
M_CrdData	将速度前瞻计算数据推送到 FIFO 中

速度前瞻是在连续插补的基础上进行运动速度的优化，可以减少速度突变带来的机械振动。让连续运动更加平滑。使用前需要实例化一块内存区以便进行运算，详情参考范例。

我们使用 Motion Assistant 来测试速度前瞻的效果，首先需要规划出如图的操场跑道：

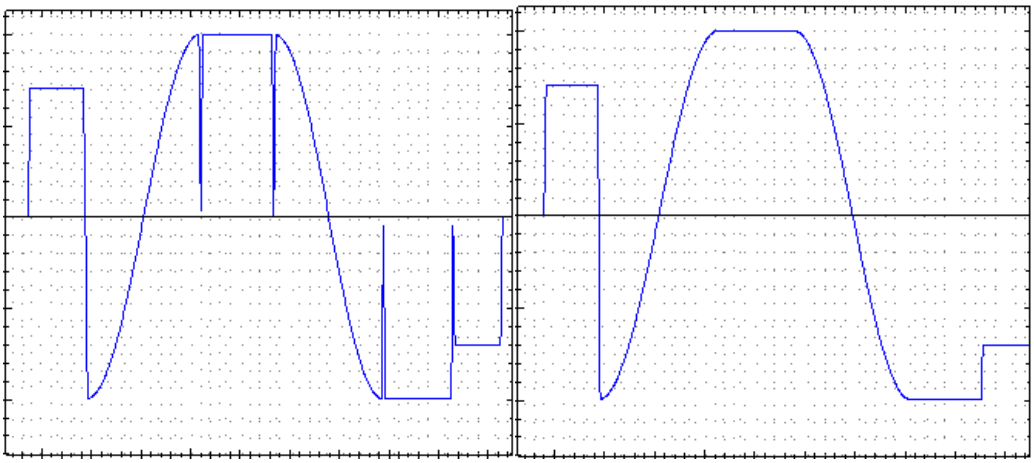


图中从（0,0）出发，运动到（100000,100000），然后顺时针运动半径为 100000 的半圆，然后运动到（300000,300000）；接着继续跑半径为 100000

的半圆，跑回（100000,100000），最终回到（0,0）。整个过程使用连续插补完成，对比开启速度前瞻与不开启时的效果。

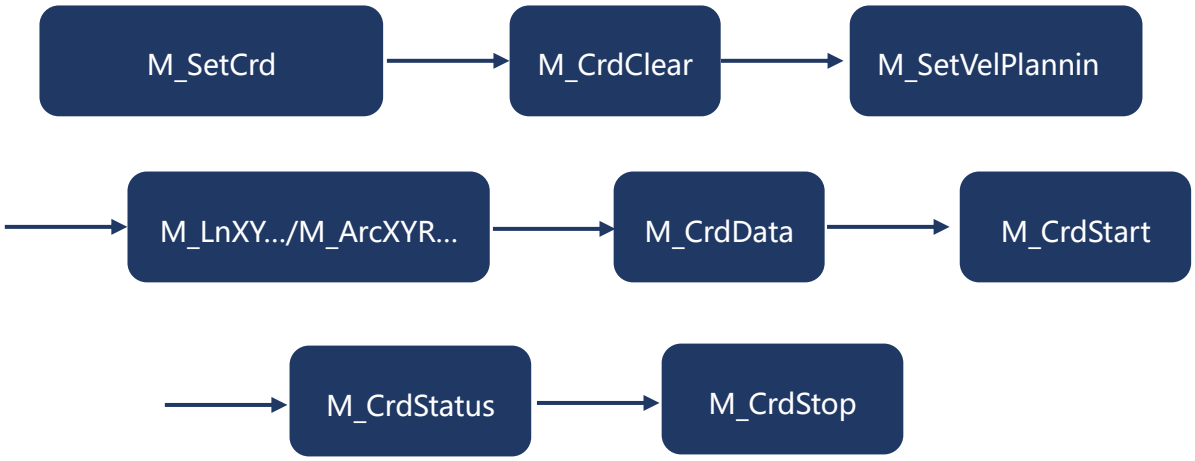
CycleTime 12100 ms　CycleTime 13502 ms

图示左侧是没有开启速度前瞻的运行时间，右侧是开启速度前瞻的运行时间。明显可见不使用速度前瞻，整个运行周期的所用时间为 12.1s，而开启速度前瞻的运行时间达到 13.502s。运行周期明显提升。这是因为系统规划速度时，把一些速度突变取消，这使得拐角时间延长从而提高了 CT。接下来看下速度曲线的对比：



如图所示，左侧是不开启速度前瞻运动时，轴 1 的速度时间曲线，右侧是开启速度前瞻的曲线。明显可以看到从圆弧到直线段有明显的速度突变，而这个突变就会导致机构的震动。右图的速度曲线由于开启了速度前瞻，没有速度突变，但是相对的整个速度曲线对比左图往右延长了，也就是运行周期边长。一般多段复杂运动会开启速度前瞻，从而降低机械振动。而简单的少点位插补运动无需使用速度前瞻，客户根据自身项目的情况自行选择是否使用。

下图为程序流程图：



例程

3.7.1 在例程 3.6.1 和 3.6.2 的基础上使用速度前瞻

```
double[] asv = new double[100000];
 //全局变量，开放一块内存区域共速度前瞻计算使用，可以不使用结构体
{
  ......
  ......
  int ret = ecat_motion.M_SetCrd (1, ref crdCfg, CardID);
  //设置插补坐标系参数到板卡
  ret = ecat_motion.M_CrdClear(1, 1, 0, CardID); //清除FIFO缓存
  short T=10; //系统拐角时间，一般经验值1-10
  double AccMax=1000000; //系统允许最大加速度
  short n=500; //速度前瞻点位计算区域
  ret = ecat_motion. M_SetVelPlanning (1,0,T,AccMax, n, ref asv[0],
  CardID);
  ......
  ......//压入点位
  ret = ecat_motion.M_CrdData(1, ref asv[0], 0, CardID);
  // 将前瞻缓存区中的数据压入控制器
  ret = lctdevice.ecat_motion.M_CrdStart(1, 0, CardID); //开始运动
  ......
  ......
}
```

M_SetVelPlanning

M_SetVelPlanning	初始化速度前瞻功能

指令说明: 初始化速度前瞻功能

指令原型: short M_SetVelPlanning (short crd, short fifo, double T, double accMax, short n, TCrdBlockData *pLookAheadBuf, short card)

参数说明:

crd	坐标系编号，可输入 1,2
fifo	先入先出缓存器，从 0 开始计数，共两个，0 和 1
T	拐角时间，单位为 cycletime，经验值是 1-10。T 越大，计算出来的终点速度越大，但却降低了加工精度；反之，提高了加工的精度，但计算出的终点速度偏低。因此要合理选择 T 值
accMax	最大加速度，单位： pulse/S/S。系统能承受的最大加速度，根据不同的机械系统和电机驱动器取值不同。
n	缓冲区大小，不小于压入点的大小，且需要小于 FIFO 容量
pLookAheadBuf	结构体数组指针首地址，存放规划计算的参数结果，不需要填入数值，只需要分配内存，可以用 INT 类型的数组空间代替，需要给到比压入点大 3~5 倍的空间
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_CrdData

M_CrdData	将速度前瞻计算数据推送到 FIFO 中

指令说明: 将速度前瞻计算数据推送到 FIFO 中

指令原型： short M_CrdData (short crd, TCrdBlockData *pLookAheadBuf, short fifo, short card)

参数说明:

crd	坐标系编号，可输入 1,2
pLookAheadBuf	结构体数组指针首地址，将之前的计算结果的首地址给出即可
fifo	先入先出缓存器，从 0 开始计数，共两个，0 和 1
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

四.IO 操作

IO 函数支持单通道输入读取和输出设置，以及 32 位输入读取和输出设置。两种方式的输入输出，方便客户更加灵活的使用。

指令	说明

M_Set_Digital_Chn_Output	设定对应通道的数字量输出
M_Set_Digital_Port_Output	设定对应端口的数字量输出(32 位输出)
M_Get_Digital_Chn_Output	获取对应通道的数字量输出
M_Get_Digital_Port_Output	获取对应端口的数字量输出(32 位输出)
M_Get_Digital_Chn_Input	获取对应通道的数字量输入
M_Get_Digital_Port_Input	获取对应端口的数字量输入(32 位输入)
M_Set_Analog_Output	设置对应模拟量通道的输出值
M_Get_Analog_Output	获取对应模拟量通道的输出值
M_Get_Analog_Input	获取对应模拟量通道的输入值
M_Get_Analog_Input_32	获取对应模拟量通道 32bit 的输入值
M_Set_Analog_Output_32	输出对应模拟量通道 32bit 的输入值
M_Set_Slave_Digital_Chn_Output	设置对应从站的数字量通道的输出状态
M_Set_Slave_Digital_Port_Output	设置对应从站的数字量端口的输出状态
M_Get_Slave_Digital_Chn_Output	获取对应从站的数字量通道的输出状态
M_Get_Slave_Digital_Port_Output	获取对应从站的数字量端口的输出状态
M_Get_Slave_Digital_Chn_Input	获取对应从站的数字量通道的输入状态
M_Get_Slave_Digital_Port_Input	获取对应从站的数字量端口的输入状态
M_Set_Slave_Analog_Output	设置从站的模拟量通道的输出值
M_Get_Slave_Analog_Output	获取从站的模拟量通道的输出值
M_Get_Slave_Analog_Input	获取从站的模拟量通道的输入值
M_Set_Slave_Analog_Output_32	获取从站的模拟量通道 32bit 的输出值

M_Get_Slave_Analog_Input_32	获取从站的模拟量通道 32bit 的输入值

1.全局 IO 读写

M_Set_Digital_Chn_Output

M_Set_Digital_Chn_Output	设定对应通道的数字量输出

指令说明: 设定对应通道的数字量输出。

指令原型: short M_Set_Digital_Chn_Output (short channel, short Value, short card)

参数说明:

channel	数字量通道，从 1 开始计数，输入输出独立
Value	设定数字量通道的数值，1 代表 on，0 代表 off
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Set_Digital_Port_Output

M_Set_Digital_Port_Output	设定对应端口的数字量输出(32 位输出)

指令说明: 设定对应端口的数字量输出(32 位输出)。

指令原型: short M_Set_Digital_Port_Output (short chnBegin, uint lValue, uint lMask, short card)

参数说明：

chnBegin	Port 输出起始的通道号
lValue	Port 输出对应的数字量值，每一位对应一个通道，32 位
lMask	Port 输出掩码，每一位对应 1 个通道，位为 1 时，对应通道能够被改写，位为 0，对应通道无法改变
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Get_Digital_Chn_Output

M_Get_Digital_Chn_Output	**获取对应通道的数字量输出**

指令说明：获取对应通道的数字量输出。

指令原型： short M_Get_Digital_Chn_Output (short channel, out short pValue, short card)

参数说明：

channel	数字量通道，从 1 开始计数，输入输出独立
pValue	获取的数字量通道数值，1 代表 on，0 代表 off
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Get_Digital_Port_Output

M_Get_Digital_Port_Output	获取对应端口的数字量输出(32 位输出)

指令说明: 获取对应端口的数字量输出(32 位输出)。

指令原型： short M_Get_Digital_Port_Output (short chnBegin, ref uint lValue, short card)

参数说明:

chnBegin	Port 输出起始的通道号
lValue	Port 输出对应的数字量值，每一位对应一个通道，32 位
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Get_Digital_Chn_Input

M_Get_Digital_Chn_Input	获取对应通道的数字量输入

指令说明: 获取对应通道的数字量输入。

指令原型： short M_Get_Digital_Chn_Input (short channel, out short pValue, short card)

参数说明:

channel	数字量通道，从 1 开始计数，输入输出独立
pValue	获取的数字量通道数值，1 代表 on，0 代表 off

card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Get_Digital_Port_Input

M_Get_Digital_Port_Input	获取对应端口的数字量输入(32 位输入)

指令说明: 获取对应端口的数字量输入(32 位输入)。

指令原型： short M_Get_Digital_Port_Input (short chnBegin, ref uint lValue, short card)

参数说明:

chnBegin	Port 输入起始的通道号
lValue	Port 输入对应的数字量值，每一位对应一个通道，32 位
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Set_Analog_Output

M_Set_Analog_Output	设置对应模拟量通道的输出值

指令说明: 设置对应模拟量通道的输出值。

指令原型： short M_Set_Analog_Output (short channel, out short pValue, short count, short card)

参数说明：

channel	模拟量通道，从 1 开始计数，输入输出独立
pValue	设置模拟量通道的数值，short 类型，-32768--32767
count	需要设置的通道数量
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Get_Analog_Output

M_Get_Analog_Output	获取对应模拟量通道的输出值

指令说明： 获取对应模拟量通道的输出值。

指令原型： short M_Get_Analog_Output (short channel, out short pValue, short count, short card)

参数说明：

channel	模拟量通道，从 1 开始计数，输入输出独立
pValue	获取模拟量通道的数值，short 类型，-32768--32767
count	需要获取的通道数量
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Get_Analog_Input

M_Get_Analog_Input	获取对应模拟量通道的输入值

指令说明: 获取对应模拟量通道的输入值。

指令原型: short M_Get_Analog_Input (short channel, out short pValue, short count, short card)

参数说明:

channel	模拟量通道，从 1 开始计数，输入输出独立
pValue	获取模拟量通道的数值，short 类型，-32768--32767
count	需要获取的通道数量
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Get_Analog_Input_32

M_Get_Analog_Input_32	获取对应模拟量通道 32bit 的输入值

指令说明: 获取对应模拟量通道 32bit 的输入值。

指令原型: short M_Get_Analog_Input_32(short channel, out uint pValue, short card)

参数说明:

channel	模拟量通道，从 1 开始计数，输入输出独立
pValue	获取模拟量通道的数值，uint 类型，0—2^32

card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Set_Analog_Output_32

M_Set_Analog_Output_32	输出对应模拟量通道 32bit 的输入值

指令说明: 输出对应模拟量通道 32bit 的输入值。

指令原型： short M_Set_Analog_Output_32(short channel, ref uint pValue, short card)

参数说明:

channel	模拟量通道，从 1 开始计数，输入输出独立
pValue	设置模拟量通道的数值，uint 类型，0—2^32
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

2.单从站 IO 读写

M_Set_Slave_Digital_Chn_Output

M_Set_Slave_Digital_Chn_Output	设置对应从站的数字量通道的输出状态

指令说明: 设置对应从站的数字量通道的输出状态。

指令原型： short M_Set_Slave_Digital_Chn_Output(short IoM, short

channel, short value, short card = 0)

参数说明:

IoM	IO 模块序号，从 1 开始计数，别名开启则为从站号
channel	数字量通道号，从 1 开始计算
value	数字量输出数值，0-不输出 1-输出
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Set_Slave_Digital_Port_Output

M_Set_Slave_Digital_Port_Output	设置对应从站的数字量端口的输出状态

指令说明: 设置对应从站的数字量端口（32 个一组）的输出状态。

指令原型： short M_Set_Slave_Digital_Port_Output(short IoM, short chnBegin, uint lValue, uint lMask, short card = 0)

参数说明:

IoM	IO 模块序号，从 1 开始计数，别名开启则为从站号
chnBegin	开始的通道号
lValue	数字量输出端口的输出数据，一个 bit 对应一个通道
lMask	数字量输出端口的输出掩码，bit 为 1，对应通道可以被改写，bit 为 0，对应通道无法被改写
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Get_Slave_Digital_Chn_Output

M_Get_Slave_Digital_Chn_Output	获取对应从站的数字量通道的输出状态

指令说明: 获取对应从站的数字量通道的输出状态。

指令原型： short M_Get_Slave_Digital_Chn_Output(short IoM, short channel, ref short value, short card = 0)

参数说明:

IoM	IO 模块序号，从 1 开始计数，别名开启则为从站号
channel	数字量通道号，从 1 开始计算
value	数字量输出数值，0-不输出 1-输出
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Get_Slave_Digital_Port_Output

M_Get_Slave_Digital_Port_Output	获取对应从站的数字量端口的输出状态

指令说明: 获取对应从站的数字量端口（32 个一组）的输出状态。

指令原型： short M_Get_Slave_Digital_Port_Output(short IoM, short chnBegin,ref uint lValue, short card = 0)

参数说明:

IoM	IO 模块序号，从 1 开始计数，别名开启则为从站号
chnBegin	开始的通道号
lValue	数字量输出端口的输出数据，一个 bit 对应一个通道
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Get_Slave_Digital_Chn_Input

M_Get_Slave_Digital_Chn_Input	获取对应从站的数字量通道的输入状态

指令说明: 获取对应从站的数字量通道的输入状态。

指 令 原 型 : short M_Get_Slave_Digital_Chn_Input(short IoM, short channel,ref short value, short card = 0)

参数说明:

IoM	IO 模块序号，从 1 开始计数，别名开启则为从站号
channel	数字量通道号，从 1 开始计算
value	数字量输出数值，0-不输出 1-输出
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Get_Slave_Digital_Port_Input

M_Get_Slave_Digital_Port_Input	获取对应从站的数字量端口的输入状

	态

指令说明: 获取对应从站的数字量端口（32 个一组）的输出状态。

指令原型： short M_Get_Slave_Digital_Port_Input(short IoM, short chnBegin,ref uint lValue, short card = 0)

参数说明:

IoM	IO 模块序号，从 1 开始计数，别名开启则为从站号
chnBegin	开始的通道号
lValue	数字量输入端口的输出数据，一个 bit 对应一个通道
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Set_Slave_Analog_Output

M_Set_Slave_Analog_Output	**设置从站的模拟量通道的输出值**

指令说明: 设置从站的模拟量通道的输出值。

指令原型： short M_Set_Slave_Analog_Output(short IoM, short channel,ref short pValue, short count = 1, short card = 0)

参数说明:

IoM	IO 模块序号，从 1 开始计数，别名开启则为从站号
channel	通道号，从 1 开始计数
pValue	模拟量输出值数组首地址
count	需要设置的通道数量

card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Get_Slave_Analog_Output

M_Get_Slave_Analog_Output	获取从站的模拟量通道的输出值

指令说明: 获取从站的模拟量通道的输出值。

指令原型： short M_Get_Slave_Analog_Output(short IoM, short channel,ref short pValue, short count = 1, short card = 0)

参数说明:

IoM	IO 模块序号，从 1 开始计数，别名开启则为从站号
channel	通道号，从 1 开始计数
pValue	模拟量输出值数组首地址
count	需要获取的通道数量
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Get_Slave_Analog_Input

M_Get_Slave_Analog_Input	获取从站的模拟量通道的输入值

指令说明: 获取从站的模拟量通道的输入值。

指令原型： short M_Get_Slave_Analog_Input(short IoM, short channel,ref

short pValue, short count = 1, short card = 0)

参数说明:

IoM	IO 模块序号，从 1 开始计数，别名开启则为从站号
channel	通道号，从 1 开始计数
pValue	模拟量输入值数组首地址
count	需要获取的通道数量
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Set_Slave_Analog_Output_32

M_Set_Slave_Analog_Output_32	**获取从站的模拟量通道 32bit 的输出值**

指令说明: 获取从站的模拟量通道 32bit 的输出值。

指 令 原 型 : short M_Set_Slave_Analog_Output_32(short IoM, short channel,ref uint pValue, short card = 0)

参数说明:

IoM	IO 模块序号，从 1 开始计数，别名开启则为从站号
channel	通道号，从 1 开始计数
pValue	模拟量输出值数组首地址
count	需要获取的通道数量
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

M_Get_Slave_Analog_Input_32

M_Get_Slave_Analog_Input_32	获取从站的模拟量通道 32bit 的输入值

指令说明: 获取从站的模拟量通道 32bit 的输入值。

指令原型: short M_Get_Slave_Analog_Input_32(short IoM, short channel, ref uint pValue, short card = 0)

参数说明:

IoM	IO 模块序号，从 1 开始计数，别名开启则为从站号
channel	通道号，从 1 开始计数
pValue	模拟量输入值数组首地址
count	需要获取的通道数量
card	卡片卡号，从 0 开始，默认按主板插槽号排序

返回值 ：详见第五章。

五.报错代码

·	使用情形	对应函数	原因	解决措施
1	连接总线	M_Connect	连接从站时从站的连接顺序被改变	需要将从站断电重启
	使能	M_Servo_On	使能超时，对应轴不处于可以使能的状态	检查对应伺服驱动器是否报警，以及设置是否正确（例如 DI 的远程使能功能需要去除）
	运动命令	M_AbsMove	EMG 信号置 ON	检查 EMG 信号是否常开，并且清除 EMG 当轴正在执行不可打断的动作时，执行了相应的运动

		M_RelMov M_Jog		指令，例如 M_STOP 中执行 JOG
	设置回零参数	M_SetHomingPrm	回零参数设置值与驱动器回读值不匹配	与驱动器手册确认回零参数范围以及单位
	IO 函数	M_Get_Digital_Chn_Input 等	数字量 IO 掉线	检查模块网口状态灯是否正常闪烁，具体排查掉线原因
2	任意函数	任意函数	固件缺少授权信息	请与供应商联系
3	回零	M_SetHomingPrm	回零参数设置值超过函数标准范围	检查参数设置数值
		M_StartHoming	轴没有使能	在 M_StartHoming 之前执行 M_Servo_On
	连接总线	M_Connect	所连接轴数超过板卡最大轴数	确认板卡版本授权信息，M04 为 4 轴卡，以此类推。
	设置参数	M_SetAxisBand 等所有设置参数函数	参数超过标准设定范围	检查参数设置数值
	IO 及运动	M_Get_Digital_Chn_Input、M_JOG 等	IO 号以及轴号填入错误	检查 IO 号和轴号是否正确
4	任意函数	任意函数	没有打开卡片	在使用其他函数前先执行 M_Open
	复位板卡	M_ResetFpga	重置 FPGA 超时	再次操作即可
5	任意总线相关函数	任意总线相关函数	选定的从站号不在从站范围内	确认从站总数以及函数的从站号形参
6	任意总线相关函数	任意总线相关函数	选定的从站设备掉线	确认选定的从站设备是否掉线，具体排查掉线原因
7	连接总线	M_Connect	连接超时	确认所连伺服驱动器是否工作再 Ethercat 模式，需重置 FPGA，可以用 M_GetConnectErr 获取错误信息，定位到具体的从站上
			FPGA 未响应	确认电脑是否进入过休眠模式，需要关闭休眠，若已进入则重启电脑
			FPGA 版本错误	查看板卡的版本信息并且与供应商联系
8	SDO 操作	SDO 相关操作，包括回零函数	SDO 操作超时	再次尝试，若依旧超时则需要联系我们具体分析
9	打开板卡	M_Open	驱动未识别到板卡	检查设备管理器内是否存在 M60，若没有请安装驱动测试，联系供应商

10	导入 ENI 文件	M_LoadEni	ENI 文件错误或者路径不正确	检查加载的 ENI 路径，和供应商联系索要最新 ENI 并且替换
	打开板卡	M_Open	板卡重复打开	检查板卡是否重复打开
11	文件操作失败	任意函数	文件操作失败	检查文件路径
12	系统资源不足	任意函数	系统资源不足	检查内存容量，以及是否以管理员运行
13	连接总线	M_Connect	未加载 Eni 文件	在 M_Connect 之前调用 M_LoadEni
14	板卡版本错误	任意函数	板卡固件不支持该函数	需要升级板卡 DSP
16	任意函数	任意函数	板卡与 PCIe 总线通讯超时	更换卡槽尝试，若无法解决请联系供应商
17	扫描总线	M_GetSlaveResource	板卡与从站通讯超时	检查板卡是否正确连接从站
19	运动命令	M_AbsMove M_RelMov M_Jog	对应轴没有进入使能状态	再运动函数之前调用 M_Servo_On，确保轴进入使能状态
20	开启别名功能	M_EnableAlias	有重复别名	将从站的别名设置为独立的
21	连接总线	M_Connect	ENI 文件中不包含所连伺服信息	用 XmlToEni 工具手动添加或者与供应商联系
23	使能操作	M_Servo_On	急停状态下使能操作	把急停按钮拔出，使用 M_ClrEmg 清除急停状态后再使能
	连接总线	M_Connect	急停状态下连接	把急停按钮拔出，使用 M_ClrEmg 清除急停状态后再连接
24	等待回零	M_WaitHomFinished	回零错误	检查回零参数是否正常，回零方向与限位是否一致
25	等待回零	M_WaitHomFinished	回零超时	检查回零参数是否合理，回零耗时是否过久