Agito-AAmotion 用户编程手册 v1.1.0

agito

www.agito-akribis.com

Member of Akribis Systems group

版本记录

版本	描述	作者	日期
1.0.0	Agito-AAMotion 用户编程手册（适用 7.0.0 之后版本 AAMotion）	Jin YaShuang	2025/5/
1.1.0	增加 AAMotionAPI 矢量运动方法 增加轴关键字 增加矢量关键字	Jin YaShuang	2025/6/

※本公司保留不定期更新的权利，根据产品硬件及软件的升级或更新迭代以及市场需求，本手册将会不定期进行内容上的更新调整，恕不另行告知，如需最新版本文档，请联系 Agito-Akribis 公司获取相应支持。

目录

简介 14

1 AAMotion 使用指南 15

1.1 创建空白项目 15

1.2 安装 NuGet 包 15

1.3 引入库文件 15

1.4 引用 DLL（必要时） 16

1.5 开始编程 16

1.6 备注 16

2 控制器初始化与连接 17

2.1 初始化控制器 17

支持的控制器类型 17

2.2 连接控制器 API 17

Connect（连接控制器） 17

Disconnect（断开控制器） 18

2.3 使用示例 18

2.4 常见问题 18

3 通用功能 20

3.1 控制器异常事件处理 20

使用 ErrorOccurred 事件捕获错误 20

3.2 AACommServer 的开启与关闭 21

Init（打开 AACommServer） 21

Shutdown（关闭 AACommServer） 21

使用说明 21

3.3 发送命令字符串方法 22

SendCommandString（发送命令） 22

SendBulkCommandString（批量发送命令） 22

示例 23

3.4 设置位置 23

SetPosition（设置位置） 23

示例代码 23

4 换向 24

4. 1 换向 API 24

Commutate（换向） 24

IsCommutated（判断换向完成） 24

4.2 换向操作示例 24

操作步骤 24

详细示例代码 24

注意事项 25
4.3 轴（AxisRef）选项列表 25
5 回零 26
5. 1 回零 API 26
Home（回零） 26
IsHoming（判断回零成功） 26
5.2 操作流程示例 26
回零操作代码示例 26
通过“HomingStat”状态判断 26
其他注意事项 27
5.3 HomingStat 状态值一览表 27
5.4 重要提醒 28
6 电机使能/失能 29
6.1 电机使能/失能 API 29
MotorOn（使能电机） 29
MotorOff（失能电机） 29
6.2 功能说明 29
使用注意事项 29
6.3 操作示例 30
例 1：启用电机 30
例 2：禁用电机 30
7 运动控制 31
7.1 运动控制 API 31
MoveRelative（重复运动） 31
MoveAbsRepetitive（重复绝对运动） 31
MoveRelRepetitive（重复相对运动） 32
Jog（Jog 运动） 32
MoveRel（相对运动） 32
MoveAbs（绝对运动） 35
使用注意事项 38
7.2 运动控制示例 38
单轴运动控制示例 38
多轴协调运动 39
运动参数说明 39
7.3 运动到位状态检测 40
InTargetStat 40
InTargetTime（到位时间） 40
InTargetTol（到位公差） 40

8 运动停止 42
8.1 运动停止 API 42
Stop（停止运动） 42
StopRep（停止重复运动） 42
StopVec（停止向量运动） 42
Abort（急停） 43
8.2 备注 43
9 控制 PCsuite IDE+ 44
9.1 程序 API 44
ProgRun（运行线程任务） 44
AutoExec（上电自动运动 IDE+） 44
ProgReset（重置程序） 44
ProgHalt（暂停线程） 44
ProgHaltAll（暂停所以线程） 45
9.2 关键字说明 45
9.3 API 使用示例 45
4.1 运行任务 46
4.2 运行主程序 46
4.3 设定自动执行 46
4.4 任务重置 46
4.5 暂停线程 46
4.6 停止所有程序 46
9.4 说明 46
10 CNC 运动 47
10.1 CNC-API 47
Begin（开始 CNC 运动） 47
Pause（暂停 CNC 运动） 47
Resume（恢复 CNC 运动） 47
ClearBuffer（清除 CNC 缓存区） 47
GroupStop（停止 CNC 运动） 47
LinearAbsolute（CNC 线性绝对运动） 48
LinearAbsoluteX（X 轴线性运动） 48
LinearAbsoluteY（Y 轴线性运动） 48
LinearAbsoluteZ（Z 轴线性运动） 49
LinearAbsoluteXY（XY 轴线性运动） 49
LinearAbsoluteXZ（XZ 轴线性运动） 49
LinearAbsoluteYZ（YZ 轴线性运动） 50
LinearAbsoluteXYZ（XYZ 轴线性运动） 50
ArcXY（XY 轴弧性运动） 50
ArcXZ（XZ 轴弧性运动） 51

ArcYZ（YZ 轴弧性运动） 52
Delay（CNC 运动延时） 52
SetMotionProfile（设置运动矢量参数） 52
SetCornerParams（设置拐角参数） 53
SetStartPositions（设置起始位置） 53
WriteDOutPort（写入离散输出） 53
GroupDOutSetBit（设置离散输出） 54
GroupDOutClearBit（清除离散输出） 54
GroupDOutToggleBit（翻转离散输出） 54
WriteGenData（写 GenData[]数组数据） 54
WriteUserParam（写 UserParam[] 数组数据） 55
GenDataWait（使用用户数组段等待） 55
UserParamWait（使用用户数组段等待） 55
SetCurrPositions（设置位置） 55
AutoCorner（自动拐角运动） 56
SetMaxVelJumpParams（设置最大速度跳跃参数） 56
SetMaxAccelParams（设置轴最大加速度） 56
MultiWriteGenData（多次写入 GenData） 57
MultiWriteUserParam（多次写入 UserParam） 57
MultiWriteGenDataWait（多次写入 GenData 并等待） 57
MultiWriteUserParamWait 58
10.2 CNC-API（CiGroup） 59
LinearAbsolute（Ci-CNC 线性绝对运动） 59
LinearAbsoluteT（T 轴线性绝对运动） 60
LinearAbsoluteXT（XT 轴线性绝对运动） 60
LinearAbsoluteYT（YT 轴线性绝对运动） 61
LinearAbsoluteZT（ZT 轴线性绝对运动） 61
LinearAbsoluteXYT（XYT 轴线性绝对运动） 62
LinearAbsoluteXZT（XZT 轴线性绝对运动） 62
LinearAbsoluteYZT（YZT 轴线性绝对运动） 62
LinearAbsoluteXYZT（XYZT 轴线性绝对运动） 63
ArcXT（XT 弧形运动） 63
ArcYT（YT 弧形运动） 64
ArcZT（ZT 弧形运动） 64
WriteICiGroupDOutPort（写入离散输出） 65
10.3 CNC-API（AGM800-CiGroup） 示例 65
10.4 CNC-API 示例 66
简介 66
示例 66
关键步骤说明 67
10.5 ArcXY 带附加圈数的注意事项 68

1. 结束速度必须设置为 0 68
2. 起点、圆心、终点必须构成有效圆弧 68
3. 延时操作注意 68
4.总结 68
10.6 MotionMode 关键字说明 69
11 数字 IO 输出 71
11.1 数字 IO-API 71
DOutClearBit（清除数字输出位） 71
DOutSetBit（设置数字输出位） 71
DOutToggleBit（翻转数字输出位） 71
GetDOutPort（得到数字输出位） 72
GetDInPort（得到数字输入位） 72
11.2 数字 IO 示例 72
11.3 注意事项 72
12 事件触发(PEG) 74
12.1 PEG-API 74
SetSingleEventPEG（设置单个 PEG 事件） 74
SetEventFixedGapPEG（设置固定间隔 PEG 事件） 74
EventEnable（使能事件触发） 75
EventDisEnable（失能事件触发） 75
12.2 eventSelect 说明 75
12.3 PEG-API 示例 76
正确示例 76
说明 77
12.4 使用注意事项 77
12.5 典型问题排查表 78
13 位置锁存(Lock) 79
13.1 Lock-API 79
SetLock（配置 Lock） 79
LockEnable（使能 Lock） 79
LockDisEnable（失能 Lock） 79
13.2 Lock-API 示例 79
13.3 Lock 功能关键字说明 80
LockEn（锁定使能） 80
LockVal（锁定值） 80
LockCntr（锁定计数器） 80
LockSrc（锁定源输入） 81
13.4 关键字使用示例 81

14 力控(Force) 82
14.1 力控-API 82
GoToForceMode（进入力控模式） 82
GoToCurrMode（进入电流模式） 82
GoToPosMode（进入位置模式） 82
SetPosition（设置目标位置值） 82
ForceCurr_SetForceCmdSource（设定力指令的来源） 83
ForceCurr_SetCurrModeSWTrigSrc（设置触发源和条件） 83
ForceCurr_SetForceTunePID（调节力控 PID 参数） 84
ForceCurr_SetMotion（调节力控 PID 参数） 84
ForceCurr_SetSlowApproache（设置缓冲逐步逼近参数） 84
ForceCurr_SetRetractMotion（设定回退运动参数） 85
ForceCurr_SetPosThForAutoSwToPosMode（设置自动切换到位置模式的阈值参数） 85
14.2 力控 API 示例 85
说明 86
14.3 注意事项 86
14.4 常见问题及解决方案 87
15 矢量运动 88
15.1 矢量运动 API 88
StopVec（停止矢量运动） 88
VectorPause（暂停矢量运动） 88
VectorContinue（恢复矢量运动） 88
VectorLinear（直线矢量运动） 88
VectorArc（弧线矢量运动） 89
SetVectorParam（配置矢量运动参数） 90
15.2 矢量运动 API 使用实例 90
1. 初始化控制器 90
2. 打开电机 91
3. 绝对位置运动 91
4. 设定矢量线性运动示例 91
5. 暂停与继续运动 92
6. 弧线运动示例 92
7. 设置运动参数 93
完整示例流程 93
16 关键字 95
16.1 轴关键字 95
Pos(位置) 96
AuxPos(辅助编码器位置) 97
PDPos (脉冲方向位置) 98

Vel (速度) 99
AuxVel (辅助速度) 101
PDVel (脉冲方向速度) 102
IaErr (A 相电流误差) 103
IbErr (B 相电流误差) 104
IdErr (Id 矢量电流控制误差) 105
IqErr (Iq 矢量电流控制误差) 106
PosRef(位置参考值) 107
VelRef(速度参考值) 108
CurrRef(电流参考) 109
IaRef(A 相电流参考) 110
IbRef(B 相电流参考) 111
IdRef(矢量 Id 电流参考) 112
IqRef(矢量 Iq 电流参考) 113
ConFlt（控制器故障码） 114
MotionStat（运动状态） 117
StatReg（状态寄存器） 118
MasterPos（齿轮主位置） 122
IndexStat（索引状态） 123
AuxIndexStat（辅助索引状态） 124
IndexPos（索引位置） 125
AuxIndexPos（辅助索引位置） 126
LimitsStat（限位状态） 127
MotorType（电机类型） 129
ContCL（连续电流限制） 131
PeakCL（峰值电流限制） 132
PeakTime（峰值时间） 135
PolePrs（磁极对数） 136
EncType（编码器反馈类型） 137
EncRes（编码器计数） 138
EncFilt（编码器数字滤波器） 139
EncDir（编码器方向） 144
AuxEncType（辅助编码器类型） 145
AuxEncFilt（辅助编码器数字滤波器） 146
AuxEncDir（辅助编码器方向反转） 147
PDEncFilt（位置反馈滤波器） 148
PDEncDir（位置反馈方向反转） 149
UsrUnits（用户单位比例） 150
AuxUsrUnits（辅助反馈位置单位） 151
PDUsrUnits（脉冲方向命令单位） 153

EmulRat（模拟编码器比率） 154
ModRev（模余模式设置） 156
MotorOn （使能/失能电机） 158
InjectType（注入信号类型） 159
ScheduleSet（调度集） 161
MotionSamples（运动样本时间数组） 162
PosErr（位置误差） 164
VelErr（速度误差） 165
DPosRef（目标位置的偏差） 166
ComtStatus（换向状态） 167
Ia（相“A”电流） 169
Ib（相“B”电流） 170
Id（直轴电流） 171
Iq（有效电流/转矩电流） 172
Va（相“A”电压） 173
Vb（相“B”电压） 174
Vc（相“C”电压） 175
Vd（直轴电压） 176
Vq（交轴电压） 177
HomingStat（回零状态） 178
MotionReason（导致最近一次运动结束的原因） 180
InTargetStat（目标到达状态） 182
MotionMode（运动模式） 184
Jerk（平滑度参数） 186
Accel（加速度） 188
Decel（减速度） 189
EmrgDecel（紧急减速度） 190
Speed（速度） 191
MaxVel（最大速度） 193
MaxAcc（最大加速度） 194
InTargetTol（到位容差） 195
InTargetTime（到位持续时间） 197
PTPKeepMoving（点到点保持运动） 198
LockEn（位置锁定启用） 199
LockCntr（位置锁定计数器） 200
LockVal（位置锁定值） 202
LockSrc（锁定功能源输入编号） 203
LockValTable（位置锁定值表） 205
LockTimeTable（位置锁定时间表） 206
UserParam（用户定义参数） 207

UserPWM（用户 PWM 控制） 208
UserPWMDiv（用户 PWM 分频参数） 210
MapEncoder（误差映射编码器） 211
MapEncoder（编码器误差映射） 213
MapLength（误差映射点数） 214
MapPosGap（误差点输入编码器间隔） 215
MapStartIndex（误差映射起始索引） 216
MapStartPos（误差映射起始位置） 217
MapTable（误差映射表） 218
MapErrOffset（误差映射偏移） 219
MapErrOffRamp（误差映射偏移斜率） 220
MapErrOnStep（误差映射步数） 221
GantryOn（龙门开启） 222
GantryAccFFW（龙门加速度前反馈） 223
GantryPosGain（龙门位置增益） 224
GantryVelGain（龙门速度增益） 225
GantryVelKi（龙门速度积分增益） 226
GantryFdbk（龙门反馈） 227
GantryOffset（龙门偏移） 228
GantryYawRef（龙门偏航参考值） 229
EventCntr（事件计数器） 230
EventSelect（事件选择） 231
EventOn（事件开启） 232
EventNextPos（事件下一个位置） 233
EventTableBeg（事件表起始位置） 234
EventTableEnd（事件表结束位置） 235
EventTableSrc（事件表源） 236
EventTableSel（事件表选择） 237
EventTable（事件表） 238
EventPulseWid（事件脉冲宽度） 239
EventType（事件类型） 240
EventBegPos（事件开始位置） 241
EventEndPos（事件结束位置） 242
EventGap（事件间隔） 244
EventTableCor（事件表校正） 246
OperationMode（操作模式） 247
ModeSwitchPos（模式切换位置） 248
PosPosFlag（位置对标志） 249
PosPosTh（位置对阈值） 250
CurrGain（电流增益） 251

CurrKi（电流积分） 253
MotorCurr（电机总电流） 255
CurrPosErrTh（位置误差阈值） 256
CurrCurrTh（电流阈值） 257
CurrCurrThDir（开环力控电流阈值方向） 258
CurrAInTh（开环力控力反馈阈值） 259
CurrCmdSrc（开环力控电流命令源） 260
CurrCmdSlope（开环力控电流命令斜率） 261
CurrCmdHTime（开环力控电流命令保持时间） 262
CurrCmdVal（开环力控电流命令值） 263
SpeedChgOn（力控慢速接近开关） 264
SpeedChgPos（力控慢速接近位置） 265
SpeedChgDir（力控慢速接近方向） 266
SpeedChgNew（力控慢速接近速度值） 267
ForceRef（力控参考值） 268
Force（力值） 269
ForceErr（力误差） 270
ForceGain（力增益） 271
ForceKi（力积分） 272
ForceKd（力微分） 273
ForceFFW（力前馈） 274
ForceVelFFW（力速度前馈） 275
ForceRefFilt（力参考滤波） 276
MaxForceErr（闭环力控最大力误差） 277
MaxForceErrOL（最大力误差开环） 278
ForcePosErrTh（闭环力控力位置误差阈值） 279
ForceAInTh（闭环力控力反馈阈值） 280
ForceCmdSrc（闭环力控力命令源） 281
ForceCmdSlope（闭环力命令斜率） 282
ForceCmdHTime（闭环力命令保持时间） 283
ForceCmdHTime（闭环力命令保持时间） 284
ForceCmdVal（闭环力命令值） 285
HomingOn（回零启动） 286
HomingDef（回零定义） 298
RetractSpeed（力控退回速度） 299
CurrDir（电流方向） 301
RevPLim（反向位置限制） 302
FwdPLim（正向位置限制） 303
MaxPosErr（最大位置误差） 304
MaxVelErr（最大速度误差） 305

HomingStep（回零步骤） 306
ComtMode（回零模式） 307
ComtAng（换向角） 311
RelTrgt（相对目标） 312
AbsTrgt（绝对目标） 313
RptWait（重复等待时间） 314
RptMode（重复模式） 315
RptCycles （执行的重复次数） 316
RptCounter（重复次数计数） 317
CurrPosTh （切换力控位置阈值） 318
CurrPosThDir（切换力控位置阈值方向） 319
BeginOnToPos （力控自动回缩开关） 320
16.2 矢量关键字 320
VecAbsTrgt（矢量运动目标位置） 321
VecAccel（矢量运动加速度） 322
VecArcCenter （矢量运动弧形运动中心点） 323
VecArcDir （矢量运动弧形运动方向） 324
VecDecel （矢量运动减速度） 326
VecdPosRef（矢量运动位置的导数） 327
VecEmrgDec（矢量运动紧急减速度） 328
VecEncRatio （矢量运动编码器分辨率） 329
VecJerk（矢量运动平滑系数） 330
VecMemberAxes （矢量运动参与轴） 331
VecMotionStat（矢量运动状态） 332
VecPause （矢量运动暂停） 333
VecPosRef （矢量位置参考值） 334
VecSpeed （矢量运动速度） 335
VecType （矢量运动类型） 336
VecNumCircles （矢量运动附加圈数） 337
VecPosFOn （矢量运动位置滤波器开关） 338
VecPosFDef（矢量运动位置滤波类型） 339
17 附录 340

简介

AAMotion 库是专为 Agito 运动控制器设计的高级 API，适用于 Windows 和 Linux 平台。该库为 AAMotion 系列运动控制器提供了丰富的接口，极大地方便了运动控制系统的开发。其主要功能包括：控制器连接、换向、回零、IDE+ 控制、单轴与多轴运动、I/O 模块管理、CNC 模式、位置锁存（Lock）、位置触发（PEG）、力控、矢量运动，以及位置事件的配置与管理等。

AAMotion 支持面向对象和面向过程两种编程方式，既可以通过类和对象进行灵活开发，也可以直接使用 AAMotionAPI 以面向对象式方式快速实现手册中的各类功能。无论是复杂系统集成还是简单应用开发，AAMotion 都能为您的运动控制项目提供高效、可靠的支持。

1 AAMotion 使用指南

AAMotion 提供面向对象的 API 接口，结构清晰，使用方便。示例代码优先采用 C#，但该 API 也支持 Python、VB6、Java、LabVIEW 等多种语言接口（C++版本请联系官方支持）。此外，用户还可以选择 ASCII 码通信方式，通过 RS232、RS485、Ethernet 或 CAN 总线实现通信。

1.1 创建空白项目

1. 打开 Visual Studio。
2. 选择“文件”>“新建”>“项目”。
3. 选择“控制台应用程序”或其他适用的项目类型（如类库、WPF 等）。
4. 命名项目并选择存储路径，点击“创建”。

1.2 安装 NuGet 包

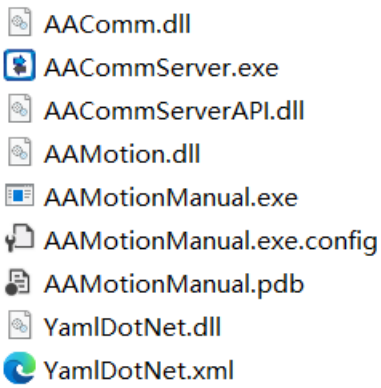
1. 在“工具”菜单中，点击“NuGet 包管理器”>“管理解决方案的 NuGet 包”。
2. 在“浏览”标签页中，搜索 **Agito.AAMotion**。
3. 找到该包后，点击“安装”。
4. 等待安装完成，确认无异常。

或，在“包管理器控制台”中运行命令：

```
Install-Package Agito.AAMotion
```

1.3 引入库文件

安装完成后，相关 DLL 文件会自动添加到项目中，或者在项目的输出目录中找到。



在项目的 bin/Debug 目录，你会看到以下 DLL 文件：

文件名	作用说明
AAComm.dll	通讯相关的 API 库，用于设备通信。

文件名	作用说明
AAMotion.dll	运动控制相关 API 库，操作机械运动。
YamDontNet.dll	依赖的基础库或扩展库（.Net 使用）

1.4 引用 DLL（必要时）

如果采用手动引入的办法（不是自动导入 NuGet 包）：
在“解决方案资源管理器”中右键点击“引用”>选择“添加引用”>选择 bin/Debug 目录中的 DLL 文件，然后确认添加。

建议：
在代码文件中加入 using 语句：

```
using AAMotion;
```

1.5 开始编程

完成以上步骤后，即可在代码中使用运动控制 AAMotionAPI（面向对象），示例：

```
// 引用命名空间
using AAMotion;
class Program
{
    static void Main(string[] args)
    {
        // 初始化控制器（示例）
        var controller = AAMotionAPI.Initialize(ControllerType.AGM800);
        // 连接控制器
        bool isConnected = AAMotionAPI.Connect(controller,”172.1.1.101”);

        if (isConnected)
        {
            Console.WriteLine("控制器已连接！");
            AAMotionAPI.MotorOn(controller, AxisRef.A); // 使能 A 轴电机
            // 后续运动控制代码
        }
        else
        {
            Console.WriteLine("连接失败，请检查硬件和网络设置。");
        }
    }
}
```

1.6 备注

- 确保 DLL 文件正确加载到项目中。
- 需要根据实际硬件连接配置调整参数。
- 可参考 API 文档，了解更多高级操作。

2 控制器初始化与连接

介绍如何使用 AAMotionAPI 进行控制器的初始化与连接操作，包括支持的控制器类型和连接方法。

2.1 初始化控制器

在进行任何操作之前，需先初始化控制器实例。

代码示例

```
public MotionController _controller =
AAMotionAPI.Initialize(ControllerType.AGM800);
```

说明

- AAMotionAPI.Initialize(ControllerType controllerType)：根据控制器类型初始化控制器实例。
- 入口参数为枚举类型 ControllerType，支持多种硬件型号。

支持的控制器类型

枚举 ControllerType 定义了所有支持的控制器类型及对应编号。

枚举值	描述	编号
AGD200	控制器型号 AGD200	5
AGD155	控制器型号 AGD155	9
AGM800	控制器型号 AGM800	10
AGD301	控制器型号 AGD301	11
AGD155EC	控制器型号 AGD155EC	12
AGD101EC	控制器型号 AGD101EC	13
AGD156EC	控制器型号 AGD156EC	14

2.2 连接控制器 API

初始化后，需连接控制器以进行下一步操作。

Connect（连接控制器）

方法一：连接默认 IP（172.1.1.101）

```
Bool Connect(MotionController controller)
```

描述	连接控制器
参数	Controller 控制器实例
返回值	连接成功返回 ture，否则返回 false
note	连接默认控制器 IP 地址 “172.1.1.101”

方法二：连接指定 IP 地址

	Bool Connect(MotionController controller, string deviceAddress)	
描述	连接控制器	
参数	Controller	控制器实例
	deviceAddress	所连接控制器的 IP 地址
返回值	连接成功返回 true，否则返回 false	
note	连接控制器（IP 地址可填的）	

Disconnect（断开控制器）

	Bool Disconnect(MotionController controller)	
描述	断开控制器	
参数	Controller	控制器实例
返回值	断开连接成功返回 true，否则 false	
note	断开当前控制器连接	

2.3 使用示例

```
// 1. 初始化控制器（使用 AGM800）
MotionController controller =
AAMotionAPI.Initialize(ControllerType.AGM800);

// 2. 连接控制器（使用默认 IP:172.1.1.101）
AAMotionAPI.Connect(controller);

// 或者，连接到特定 IP 地址
// AAMotionAPI.Connect(controller, "192.168.0.100");
```

2.4 常见问题

Q：如果连接失败，应该如何处理？

A：确保控制器设备已正确连接网络，IP 地址正确无误，且控制器已开机。可以尝试重新初始化或检测网络连通性。

Q：如何切换控制器或更改 IP 地址？

A：在 PCSuite-配置-通讯。更改之后保存到闪存，设备重新上电就是使用新的 IP 地址

3 通用功能

3.1 控制器异常事件处理

使用 ErrorOccurred 事件捕获错误

控制器在运动或运行时发生错误，或者是在使用 AAMotionAPI 发送命令失败使，内部会通过 ErrorOccurred 事件向应用程序传递错误码及错误信息。示例代码：

```
{
    InitializeComponent();
    //实例化控制器
public MotionController _controller =
AAMotionAPI.Initialize(ControllerType.AGM800);

    // 事件绑定：设备错误信息
    controller.ErrorOccurred += (errorCode, MsgSent, errorMsg) =>
    {
        ShowLog($"Message Sent: {MsgSent}");
        ShowLog($"Error Code: {errorCode}");
        ShowLog($"Error Message: {errorMsg}");
    };
}
```

错误处理建议

- **实时响应**：根据 errorCode 和 errorMsg 立即采取措施（如断电、安全停机等）。
- **状态监测**：结合运动状态检测，判断是否有异常发生。
- **错误码查看**：ErrLog、ConFlt

错误类型

通常有 2 种类型的错误。
解释器错误
当您发送命令时，控制器可能会拒绝
例如，发送的命令：BMotorOn = 1
要捕获这些类型的错误，您可以向 ctrl 附加回调。错误发生
Ctrl 键。ErrorOccurred += （errCode， msgSent， errorMsg） =>
{
MessageBox.Show（errCode.ToString（） ）；
MessageBox.Show（msgSent）；
MessageBox.Show（errorMsg）；
}；
您将获得：
错误代码： 159
msgSent 的 BMotorOn=1
errorMsg 中：
错误：159
发送命令：BMotorOn=1

错误：此请求仅对活动（已连接）Central-i 端口有效

控制器故障
当电机控制相关故障发生时。
例如，电机电流过高。
用户需要轮询 ConFlt 并检查它何时不为 0

错误使用范围

1．特殊错误类型处理
- 同步错误（SyncErr）：前缀为 ERR Sync:，通常由多线程并发调用 SendReceive 导致（SendReceive 非线程安全）。
- PC Suite 错误（PcSuiteErr）：前缀为 PC Suite: ERR，表示 API 层检测到的错误（如消息格式非法，未发送至控制器）。
- Boot 模式错误（BootErr）：前缀为 BOOT ERR，表示控制器处于 Boot 模式时返回的错误。

ErrorOccurred 完整覆盖了从命令发送到响应接收的全流程错误捕获，通过 AAComm 实现了错误码的解析与格式化，能够准确定位错误类型（通讯/控制器/参数问题）及影响范围

3.2 AACommServer 的开启与关闭

Init（打开 AACommServer）

	bool Init(MotionController controller)	
描述	打开 AACommServer（Agito 通讯应用）	
参数	MotionController controller	控制器实例
返回值	true：指令发送成功，操作执行完成。	
	false：指令发送失败或通信异常	
备注	用于初始化控制器通信环境，实际上会自动启动 **AACommServer**，实现 PC 端与控制器的通信连接	

Shutdown（关闭 AACommServer）

	bool Shutdown(MotionController controller)	
描述	关闭 AACommServer	
参数	MotionController controller	控制器实例
返回值	true：指令发送成功，操作执行完成。	
	false：指令发送失败或通信异常	
备注	用于关闭控制器通信环境，停止 **AACommServer** 的运行，断开 PC 端与控制器的通信。	

使用说明

- **自动管理**：在调用 connect 进行连接时，**AACommServer** 会自动启动，无需用户手动调用 Init()。

- **手动控制**：如果在特定场景需要控制通信的开启与关闭，可以在连接前调用 `Init()`，在完成通信后调用 `Shutdown()` 以确保资源释放。

- **`Init()` 和 `Shutdown()`**是对控制器通信底层资源的管理方法，确保在不需要通信时及时关闭，以节省资源。
- 在调用 `Connect()` 时，若 AACommServer（底层通讯进程）已启动（由 `Init()` 启动或之前已开启），则无需重复启动。

3.3 发送命令字符串方法

SendCommandString（发送命令）

	bool SendCommandString(MotionController controller, string rawCommand, out string response)	
描述	将指定的命令字符串发送到控制器，并等待响应。适用于普通控制命令的传输。	
参数	MotionController controller	控制器实例
	string rawCommand	待发送的命令字符串（如控制指令、查询指令等）
	out string response	输出参数，用于接收控制器返回的响应信息
返回值	true：命令成功传输并获得响应，未检测到错误。	
	false：通信过程中发生错误，响应内容包含错误信息。	
备注	传输单个控制指令（如启动、停止、状态查询等）。	
	适用于快速、短指令的通信。	

SendBulkCommandString（批量发送命令）

	bool SendBulkCommandString(MotionController controller, string rawCommand, out string response)	
描述	将批量（Bulk）命令字符串发送到控制器，支持更大指令集或复合命令，等待响应。	
参数	MotionController controller	控制器实例
	string rawCommand	待发送的批量命令字符串
	out string response	输出参数，用于接收控制器返回的响应信息
返回值	true：批量命令成功传输，响应正常。	
	false：通信发生错误，响应内容在 ErrorOccurred 包含错误细节。	
备注	需要一次性传输多条指令或大块数据，最多 50 条指令。	

系统需要批量配置时使用。

示例

```
string response;
bool success = SendCommandString(controller, "ASPeed", out response);
if (success)
{
    Console.WriteLine("命令成功，响应：" + response);
}
```

3.4 设置位置

SetPosition（设置位置）

	bool SetPosition(MotionController controller, AxisRef axis, int value)	
描述	用于为主轴设置新的位置读数（位置参考值）。可以自主分配一个新的位置数值，影响位置控制和后续运动参考。	
参数	MotionController controller	控制器实例
	AxisRef axis	轴引用
	int value	要设定的新位置值
返回值	true：命令成功传输，响应正常。	
	false：通信发生错误，ErrorOccurred 响应内容包含错误细节。	
备注	不能在以下条件下调用 SetPosition，否则控制器会返回错误码： 1. 轴正在运动时。 2. 错误映射（Error Mapping）已激活。 3. 自动增益（自动惯性）已激活。 4. SetPosition 的值超出软件位置限制范围（RevPLim 到 FwdPLim）。 5. 伺服已使能且输入整形已激活。 - SetPosition 将更新相关内部变量，以确保平滑操作，无突兀变化。 - 仅在特定维护或调试场景下使用，避免影响正常运动控制	

示例代码

```
// 1. 初始化控制器
MotionController controller =
AAMotionAPI.Initialize(ControllerType.AGM800);

// 2. 将 A 轴位置重置为 0
AAMotionAPI.SetPosition(controller, AxisRef.A, 0);
```

4 换向

换向（Commutation）用于控制运动轴的方向切换，是运动控制中常见的操作。通过 API，可以方便地对指定轴执行换向指令，并确认换向状态。

4.1 换向 API

Commutate（换向）

	Void Commutate(MotionController controller , AxisRef axisRef, int? waitTime = null)	
描述	对指定轴执行换向操作。设定等待时间（可选）。	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int? waitTime	等待时间（毫秒，可选）
返回值	无	

IsCommutated（判断换向完成）

	bool IsCommutated(MotionController controller , AxisRef axisRef)	
描述	检查轴是否已完成换向。	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
返回值	true：已换向； false：未换向	

4.2 换向操作示例

操作步骤

1. 调用 `Commutate()` 方法执行换向。
2. 使用 `IsCommutated()` 方法确认换向完成。

详细示例代码

```
// 指定轴进行换向操作，等待 5000 毫秒（5 秒）
AAMotionAPI.Commutate(controller, AxisRef.D, 5000);

// 检查轴是否已完成换向
bool isCommutated = AAMotionAPI.IsCommutated(controller, AxisRef.D);

// 判断换向状态
if (isCommutated)
{
    Console.WriteLine("换向成功，轴已完成换向。");
```

```
    // 后续可以添加其他操作
}
else
{
    Console.WriteLine("换向未完成，是否需要重新执行。");
    // 选择重新换向或提示用户
}
```

注意事项

- 在调用 `Commutate()`后，应调用 `IsCommutated()`确认换向是否完成，确保操作成功。
- 如果换向未完成，可考虑增加等待时间或重新执行操作。
- 确保控制器和轴状态正常，避免因设备状态异常导致换向失败。
- “确保在 PCSUite 可以正确的进行换向，将正确的换向操作可以保存到 Flash 中”

4.3 轴（AxisRef）选项列表

轴标识	枚举值	描述
A	1	轴 A
B	2	轴 B
C	4	轴 C
D	8	轴 D
E	16	轴 E
F	32	轴 F
G	64	轴 G
H	128	轴 H
I	256	轴 I
J	512	轴 J
K	1024	轴 K
L	2048	轴 L

5 回零

回零操作是运动控制的基础前置步骤，通过将机械坐标系与物理原点对齐

5.1 回零 API

Home（回零）

	Void Home(MotionController controller, AxisRef axis, string filePath)	
描述	轴回零操作，指定回零文件路径	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	string filePath)	回零参数文件路径（由 PCSuite 去进行导出）
返回值	无	

IsHoming（判断回零成功）

	bool IsHomed(MotionController controller, AxisRef axis)	
描述	判断回零是否成功	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
返回值	true：回零成功； false：回零失败	

5.2 操作流程示例

回零操作代码示例

```
// 设定回零文件路径
string homeFilePath = "你的回零文件路径"; // 替换为实际文件路径

// 执行回零
AAMotionAPI.Home(controller, AxisRef.D, homeFilePath);
// 等待一段时间后检查
bool isHomed = AAMotionAPI.IsHomed(controller, AxisRef.D);
if (isHomed)
{
    Console.WriteLine("回零成功！");
}
else
{
    Console.WriteLine("回零未完成或失败。");
}
```

通过“HomingStat”状态判断

除了调用 IsHomed() 方法外，你还可以通过读取“HomingStat”关键字的值，判断回零是否成功。

示例：

```
string response;
string command = "AHomingStat"; // A 轴回零状态
bool success = AAMotionAPI.SendCommandString(controller, command, out
response);
if (success) {
// 判断响应内容是否包含"100"
if (response.Contains("100"))
 { Console.WriteLine("回零成功！"); }
else { Console.WriteLine("回零未成功，响应内容： " + response); } }
else { Console.WriteLine("命令发送失败！");}
```

其他注意事项

- 确保回零文件路径正确，文件存在且格式符合规范。
- 可以在回零后立即确认 IsHoming()返回值，确保回零成功再进行后续操作。

5.3 HomingStat 状态值一览表

状态值	说明	详细解释	备注
0	未执行	尚未开始实际回零操作，可能处于等待或未命令状态。	可以通过状态查询或命令确认。
100	成功完成	轴已成功回零，回零操作已完成。	可进行后续的正常操作。
-1	参数错误导致失败	在 h HomingDef 数组参数中的参数设置有误，回零过程被中止（参数在每个步骤开始时验证）。	检查参数设置是否正确，参数范围是否适配设备。
-2	超时导致失败	某个回零步骤在预设超时时未完成，导致回零操作中止。每个步骤具有超时时间设定，超时后会中断操作。	可优化超时时间或保证设备响应时间。
-3	电机意外断电或禁用造成失败	在某步骤中检测到电机掉使能状态或意外断电，导致回零中止。参照 MotorReason 指令中状态。	检查硬件电源和驱动状态，确保电机正常工作。
-4	运动原因导致失败	在某个步骤，运动结束原因（如索引、检测到目标位置）与预期不同，导致回零中止。	检查机械路径与传感器状态。
-5	步骤类型错误导致失败	设定的步骤类型在配置中不可识别或不支持，回零中止。	重新配置正确的步骤类型。
-6	运动中无	轴仍有运动在进行中，未完成前不能启动回	等待运动完成后再次

状态值	说明	详细解释	备注
	法设置回零	零。	尝试。
-7	步骤数超出范围	定义的回零步骤数量超出允许范围，或到达最后一阶段但未定义为结束状态。	调整参数，确保满足范围和定义。
-8	机械限制影响导致失败	机械或安全限制被激活（如限位开关），阻止回零动作。	解锁限制条件，确保安全后重试。
-9	设置位置失败	试图设置新位置或目标位置失败，参考 SetPosition 指令的相关说明。	校验目标位置参数和配置。
-10	运动模式超界或不支持	所选择的运动模式（或齿轮模式）不存在或不允许。	选择支持的运动类型或修改配置。
-11	误差补偿超出范围	设置的误差补偿参数不在设备允许范围内。	调整参数到允许范围内。
-12	自动换相未完成，操作失败	自动换相操作未完成，影响后续回零。	先完成自动换相后再进行回零。

说明：

- 值“100”明确表示回零操作成功完成。
- **负值**表明操作失败，详细原因见“详细解释”部分。
- 值“0”表示还未开始回零或未检测到状态。

5.4 重要提醒

- 在使用 `HomingStat` 判断回零状态时，推荐结合 `SendCommandString` 返回的值，特别是当返回“100”时，说明回零已完成。
- 若状态为负值或未成功，建议检查参数配置、硬件状态、运动路径等，确保条件满足
- 操作前应确保机械安全和设备确认就绪。

6 电机使能/失能

通过 API 接口，可以对机械设备的电机进行使能（MotorOn）或失能（MotorOff）操作。这一功能用于启动或停止电机的驱动，使机械运动控制更加灵活和安全。

6.1 电机使能/失能 API

MotorOn（使能电机）

	void MotorOn(MotionController controller, AxisRef axis)	
描述	启用电机，让目标轴开始受控，电机驱动力激活	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
返回值	无	

MotorOff（失能电机）

	void MotorOff(MotionController controller, AxisRef axis)	
描述	禁用电机，关闭电机，停止驱动力，电源不再施加	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
返回值	无	

6.2 功能说明

- **启用电机（MotorOn）：**
  - 将 `MotorOn` 属性设置为 `1`，电源被施加到电机，驱动器开始工作。
  - 允许发出运动命令，执行位置控制、回零等操作。
- **禁用电机（MotorOff）：**
  - 将 MotorOn 属性设置为 0，停止电机驱动。
  - 电源断开，电机不受驱控，确保安全。

使用注意事项

- **安全第一：**在启用电机前，应确保机械安全，无障碍，避免运动造成伤害。
- **故障自动禁用：**如果驱动器检测到故障（如过温、过载等），会自动禁用电机（`ConFlt` 状态），即使用户手动启用，电机也不会正常工作，需先解决故障。
- **重新启用电机：**
  - 如果驱动器中存在故障（ConFlt 不为 0），在再次启用前应消除故障状态。
  - 重新调用 MotorOn() 后，ConFlt 会被清除，但若硬件状态未恢复（如过热未散去），电机会仍可能保持禁用状态。

- **参数限制：**
    - 某些参数的修改或设置仅允许在电机禁用状态下进行。

6.3 操作示例

例 1：启用电机

```
// 启用轴 1 的电机
AAMotionAPI.MotorOn(controller, AxisRef.A);
Console.WriteLine("控制器 A 轴已启用。");
```

例 2：禁用电机

```
// 禁用轴 1 的电机
AAMotionAPI.MotorOff(controller, AxisRef.A);
Console.WriteLine("控制器 A 已禁用。")
```

7 运动控制

AAMotion 库支持多种运动控制模式，满足不同自动化需求：

运动类型	特点
Jog 运动	持续运动，直到停止
PTP 绝对运动	移到指定绝对位置
PTP 相对运动	从当前位置偏移指定距离
重复运动	在两个点间循环运动

7.1 运动控制 API

MoveRelative（重复运动）

	void MoveRelative(MotionController controller, AxisRef axis, int distance, int? speed = null, int? accel = null, int? decel = null)	
描述	轴相对运动，即移动目标轴相对于当前位置的距离	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int distance	移动距离（正为正向，负为反向）
	int? speed（可选）	运动速度
	int? accel（可选）	加速度
	int? decel（可选）	减速度
返回值	无返回值，执行后轴开始相对运动	

MoveAbsRepetitive（重复绝对运动）

	void MoveAbsRepetitive(MotionController controller, AxisRef axis, int position, int msDwellTime, int? speed = null, int? accel = null, int? decel = null)	
描述	轴重复绝对位置运动	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int distance	目标位置（绝对坐标）
	int msDwellTime	每次位置停留毫秒数
	int? speed（可选）	运动速度
	int? accel（可选）	加速度
	int? decel（可选）	减速度

返回值	无返回值，目标轴按设定位置反复运动，停留指定时间

MoveRelRepetitive（重复相对运动）

	void MoveRelRepetitive(MotionController controller, AxisRef axis, int position, int msDwellTime, int? speed = null, int? accel = null, int? decel = null)	
描述	轴重复相对运动	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int distance	目标位置（绝对坐标）
	int msDwellTime	每次位置停留毫秒数
	int? speed（可选）	运动速度
	int? accel（可选）	加速度
	int? decel（可选）	减速度
返回值	无返回值，目标轴反复沿距离运动，停留时间指定	

Jog（Jog 运动）

	Void Jog(MotionController controller, AxisRef axis, int velocity)	
描述	轴连续运动（Jog 模式）	
参数	Controller	控制器实例
	deviceAddress	所连接控制器的 IP 地址
	int velocity	运动速度（正负表示不同方向）
返回值	无返回值，开始 Jog 运动，需后续停止	

MoveRel（相对运动）

	void MoveRel(MotionController controller,int accel, int speed, int decel, ISingleAxis a, int aPos, ISingleAxis b, int bPos)	
描述	两轴相对运动	
参数	MotionController controller	控制器实例
	int accel	加速度
	int decel	减速度
	ISingleAxis a	A 轴标识
	int aPos	A 轴运动目标位置
	ISingleAxis b	B 轴标识
	int bPos	B 轴运动目标位置
返回值	返回 bool，表示运动是否成功开始	

	void MoveRel(MotionController controller,int accel, int speed, int decel, ISingleAxis a, int aPos, ISingleAxis b, int bPos, ISingleAxis c, int cPos)	
描述	三轴相对运动	
参数	MotionController controller	控制器实例
	int accel	加速度
	int decel	减速度
	ISingleAxis a	A 轴标识
	int aPos	A 轴运动目标位置
	ISingleAxis b	B 轴标识
	int bPos	B 轴运动目标位置
	ISingleAxis c	C 轴标识
	int cPos	C 轴运动目标位置
返回值	返回 bool，表示运动是否成功开始	

	void MoveRel(MotionController controller,int accel, int speed, int decel, ISingleAxis a, int aPos, ISingleAxis b, int bPos, ISingleAxis c, int cPos)	
描述	三轴相对运动	
参数	MotionController controller	控制器实例
	int accel	加速度
	int decel	减速度
	ISingleAxis a	A 轴标识
	int aPos	A 轴运动目标位置
	ISingleAxis b	B 轴标识
	int bPos	B 轴运动目标位置
	ISingleAxis c	C 轴标识
	int cPos	C 轴运动目标位置
返回值	返回 bool，表示运动是否成功开始	

	void MoveRel(MotionController controller,int accel, int speed, int decel, ISingleAxis a, int aPos, ISingleAxis b, int bPos, ISingleAxis c, int cPos,ISingleAxis d, int dPos)	
描述	四轴相对运动	
参数	MotionController controller	控制器实例
	int accel	加速度
	int decel	减速度
	ISingleAxis a	A 轴标识

	int aPos	A 轴运动目标位置
	ISingleAxis b	B 轴标识
	int bPos	B 轴运动目标位置
	ISingleAxis c	C 轴标识
	int cPos	C 轴运动目标位置
	ISingleAxis d	D 轴标识
	int dPos	D 轴运动目标位置
返回值	返回 bool，表示运动是否成功开始	

	void MoveRel(MotionController controller,int accel, int speed, int decel, ISingleAxis a, int aPos, ISingleAxis b, int bPos, ISingleAxis c, int cPos,ISingleAxis d, int dPos，ISingleAxis e, int ePos)
描述	五轴相对运动

参数	MotionController controller	控制器实例
	int accel	加速度
	int decel	减速度
	ISingleAxis a	A 轴标识
	int aPos	A 轴运动目标位置
	ISingleAxis b	B 轴标识
	int bPos	B 轴运动目标位置
	ISingleAxis c	C 轴标识
	int cPos	C 轴运动目标位置
	ISingleAxis d	D 轴标识
	int dPos	D 轴运动目标位置
	ISingleAxis e	E 轴标识
	int EPos	E 轴运动目标位置
返回值	返回 bool，表示运动是否成功开始	

	void MoveRel(MotionController controller,int accel, int speed, int decel, ISingleAxis a, int aPos, ISingleAxis b, int bPos, ISingleAxis c, int cPos,ISingleAxis d, int dPos，ISingleAxis e, int ePos,ISingleAxis f, int fPos)
描述	六轴相对运动

参数	MotionController controller	控制器实例
	int accel	加速度
	int decel	减速度

	ISingleAxis a	A 轴标识
	int aPos	A 轴运动目标位置
	ISingleAxis b	B 轴标识
	int bPos	B 轴运动目标位置
	ISingleAxis c	C 轴标识
	int cPos	C 轴运动目标位置
	ISingleAxis d	D 轴标识
	int dPos	D 轴运动目标位置
	ISingleAxis e	E 轴标识
	int ePos	E 轴运动目标位置
	ISingleAxis f	F 轴标识
	int fPos	F 轴运动目标位置
返回值	返回 bool，表示运动是否成功开始	

MoveAbs（绝对运动）

	void MoveAbs(MotionController controller, AxisRef axis, int position, int msDwellTime, int? speed = null, int? accel = null, int? decel = null)
描述	轴绝对运动，即移动到指定绝对位置
参数	MotionController controller 控制器实例
	AxisRef axisRef 轴的标识
	int distance 目标位置（绝对坐标）
	int? speed（可选） 运动速度
	int? accel（可选） 加速度
	int? decel（可选） 减速度
返回值	无返回值，执行后轴移动到目标位置

	void MoveAbs(MotionController controller,int accel, int speed, int decel, ISingleAxis a, int aPos, ISingleAxis b, int bPos)
描述	两轴绝对运动
参数	MotionController controller 控制器实例
	int accel 加速度
	int decel 减速度
	ISingleAxis a A 轴标识
	int aPos A 轴运动目标位置
	ISingleAxis b B 轴标识

	int bPos	B 轴运动目标位置
返回值	返回 bool，表示运动是否成功开始	

	void MoveAbs(MotionController controller,int accel, int speed, int decel, ISingleAxis a, int aPos, ISingleAxis b, int bPos, ISingleAxis c, int cPos)

描述	三轴绝对运动	
参数	MotionController controller	控制器实例
	int accel	加速度
	int decel	减速度
	ISingleAxis a	A 轴标识
	int aPos	A 轴运动目标位置
	ISingleAxis b	B 轴标识
	int bPos	B 轴运动目标位置
	ISingleAxis c	C 轴标识
	int cPos	C 轴运动目标位置
返回值	返回 bool，表示运动是否成功开始	

	void MoveAbs(MotionController controller,int accel, int speed, int decel, ISingleAxis a, int aPos, ISingleAxis b, int bPos, ISingleAxis c, int cPos)

描述	两轴绝对运动	
参数	MotionController controller	控制器实例
	int accel	加速度
	int decel	减速度
	ISingleAxis a	A 轴标识
	int aPos	A 轴运动目标位置
	ISingleAxis b	B 轴标识
	int bPos	B 轴运动目标位置
	ISingleAxis c	C 轴标识
	int cPos	C 轴运动目标位置
返回值	返回 bool，表示运动是否成功开始	

	void MoveAbs(MotionController controller,int accel, int speed, int decel, ISingleAxis a, int aPos, ISingleAxis b, int bPos, ISingleAxis c, int cPos,ISingleAxis d, int dPos)

描述	四轴绝对运动	
参数	MotionController controller	控制器实例

	int accel	加速度
	int decel	减速度
	ISingleAxis a	A 轴标识
	int aPos	A 轴运动目标位置
	ISingleAxis b	B 轴标识
	int bPos	B 轴运动目标位置
	ISingleAxis c	C 轴标识
	int cPos	C 轴运动目标位置
	ISingleAxis d	D 轴标识
	int dPos	D 轴运动目标位置
返回值	返回 bool，表示运动是否成功开始	

	void MoveAbs(MotionController controller,int accel, int speed, int decel, ISingleAxis a, int aPos, ISingleAxis b, int bPos, ISingleAxis c, int cPos,ISingleAxis d, int dPos，ISingleAxis e, int ePos)
描述	五轴绝对运动

参数	MotionController controller	控制器实例
	int accel	加速度
	int decel	减速度
	ISingleAxis a	A 轴标识
	int aPos	A 轴运动目标位置
	ISingleAxis b	B 轴标识
	int bPos	B 轴运动目标位置
	ISingleAxis c	C 轴标识
	int cPos	C 轴运动目标位置
	ISingleAxis d	D 轴标识
	int dPos	D 轴运动目标位置
	ISingleAxis e	E 轴标识
	int EPos	E 轴运动目标位置
返回值	返回 bool，表示运动是否成功开始	

	void MoveAbs(MotionController controller,int accel, int speed, int decel, ISingleAxis a, int aPos, ISingleAxis b, int bPos, ISingleAxis c, int cPos,ISingleAxis d, int dPos，ISingleAxis e, int ePos,ISingleAxis f, int fPos)
描述	六轴绝对运动

参数	MotionController controller	控制器实例
	int accel	加速度
	int decel	减速度
	ISingleAxis a	A 轴标识
	int aPos	A 轴运动目标位置
	ISingleAxis b	B 轴标识
	int bPos	B 轴运动目标位置
	ISingleAxis c	C 轴标识
	int cPos	C 轴运动目标位置
	ISingleAxis d	D 轴标识
	int dPos	D 轴运动目标位置
	ISingleAxis e	E 轴标识
	int ePos	E 轴运动目标位置
	ISingleAxis f	F 轴标识
	int fPos	F 轴运动目标位置
返回值	返回 bool，表示运动是否成功开始	

使用注意事项

- 运动前确认所有 `ISingleAxis` 轴已启用电机（`MotorOn` = 1）
- 目标位置数组长度应匹配轴数组长度
- 运动参数（速度、加速度、减速度）应在设备和轴允许范围内
- 在运动过程中可监控状态控制停止

7.2 运动控制示例

单轴运动控制示例

Jog 运动

实现方式：

```
// 启动 Jog 运动
AAMotionAPI.Jog(controller, AxisRef.A, 5000); // 正向，速度 5000 count

AAMotionAPI.Stop(controller, AxisRef.A);// 停止 Jog 运动
```

PTP 绝对运动

移动到指定位置：

```
// 无参数，使用轴的默认速度、加减速
```

```
AAMotionAPI.MoveAbs(controller, AxisRef.B, 100000);
// 指定运动参数
AAMotionAPI.MoveAbs(controller, AxisRef.C, 150000, speed: 20000, accel:
10000, decel: 8000);
```

PTP 相对运动

相对于当前位置偏移：

```
// 无参数，使用默认运动参数
AAMotionAPI.MoveRel(controller, AxisRef.D, 5000);

// 指定运动参数
AAMotionAPI.MoveRel(controller, AxisRef.E, -3000, speed: 15000, accel:
5000);
```

重复运动

在两个位置间循环往返：

```
// 绝对位置重复运动
AAMotionAPI.MoveAbsRepetitive(controller, AxisRef.F, 10000, 500);

// 相对位置往复运动
AAMotionAPI.MoveRelRepetitive(controller, AxisRef.F, 10000, 500);
```

多轴协调运动

多轴 PTP 运动

同步控制多轴运动：

```
// 两轴同步绝对运动
AAMotionAPI.MoveAbs(accel: 10000, speed: 20000, decel: 8000,
                    axisA, 100000, axisB, 150000);

// 三轴同步相对运动
AAMotionAPI.MoveRel(accel: 12000, speed: 18000, decel: 10000,
                    axisX, 5000, axisY, 8000, axisZ, 2000);
```

运动参数说明

参数	单位	描述	备注
speed	Count/秒	运动速度	不能超过轴的最大速度 MaxVel
accel	Count/秒²	加速度	不能超过 MaxAcc
decel	Count/秒²	减速度	建议等于加速度

7.3 运动到位状态检测

为了确保运动已完成且轴已到达目标位置，可以通过轮询检测 `InTargetStat` 属性。`InTargetStat` 的值为 4 时，表示轴已到达目标位置。

InTargetStat

`InTargetStat` 属性用于指示轴是否到达预期的目标位置，它的状态值代表不同的运动状态：

值	状态	说明
0	电机关闭	轴驱动已关闭，不在运动状态
1	闲置（伺服状态使能）	轴伺服已使能，未运动
2	运动中	轴正向或反向正在运动
3	等待 InTargetTime 结束	运动已到达目标位置，等待 InTargetTime 时间完成
4	目标已到达	轴已到达目标位置，运动完成

示例

在检测某个轴是否到位时，通常轮询该值：

```
// 先发起绝对运动，将轴移动到位置 0（示例）
AAMotionAPI.MoveAbs(controller, AxisRef.A, 100000, 1000); // 目标位置 100000，速度 1000
while (InTargetStat != 4); // 直到为 4，表示到位
```

说明

- 到位判断：当 InTargetStat 的值为 4 时，表示轴已到达预设目标位置，可视为运动成功完成。
- 等待目标到达：在某些运动场景中，InTargetStat 会在到达后等待用户设置的 InTargetTime（毫秒）时间段，确保运动稳定
- 上述代码会持续检测 InTargetStat，直到其值为 4，表示目标已完全到达。

InTargetTime（到位时间）

InTargetTime 是电机应在目标公差范围内的时间，单位为毫秒，以确定目标是否已达到。如果位置误差在用户单位中小于 InTargetTol 并且持续至少 InTargetTime 毫秒，那么 InTargetStat 将接收到一个值，表示目标已达到。

InTargetTol（到位公差）

InTargetTol 是“目标内”公差。如果位置误差在用户单位中小于 InTargetTol 并且持续至少 InTargetTime 毫秒，那么 InTargetStat 将接收到一个值，表示目标已达到。

8 运动停止

在运动控制中，不同的停止命令满足不同的控制需求。以下介绍常用停止方法及其适用场景。

8.1 运动停止 API

Stop（停止运动）

```
bool Stop(MotionController controller, AxisRef axis)
```

描述	停止运动	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
返回值	无	
说明	作用：按照在 Decel 参数中定义的减速度，逐渐减速停止运动。 特点：不会立即中断当前运动，而是平滑减速，避免冲击。 注意：如果当前没有运动进行，调用此命令不会有效果。 应用场景：在安全允许的情况下，平滑、安全地终止运动。	

StopRep（停止重复运动）

```
bool StopRep(MotionController controller, AxisRef axis)
```

描述	停止重复运动	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
返回值	无	
说明	作用：停止正在执行的重复运动（如循环动作）。 特点： 仅停止“重复运动”，如果没有重复运动进行，命令无效果。 此方法不会立即停止当前运动，而是在当前运动结束后在起点停止（类似逐路停止的效果）。 应用场景：在执行多次重复运动（如振荡、循环任务）时，用于中断下一轮循环。	

StopVec（停止向量运动）

```
bool StopVec(MotionController controller, AxisRef axis)
```

描述	停止矢量运动	
参数	MotionController controller	控制器实例

	AxisRef axisRef	轴的标识
返回值	无	

作用：用于停止一个运动向量（多轴联合运动）。

特点： 作用类似 Stop()，但针对多轴向量运动。适用于多轴同步运动的停止。

应用场景：多轴联动运动中，需要安全停止所有运动。

Abort（急停）

```
bool Abort(MotionController controller, AxisRef axis)
```

描述	急停	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
返回值	无	

作用：立即中断当前运动，无预警，快速停止。

特点： 不会缓冲减速，直接中止运动。在没有运动时调用也无效果。这是最激烈的停止方式，适用于紧急情况（急停）。

应用场景：紧急状况需要立即停止全部或局部运动，保证安全。

8.2 备注

- 在正常运动中，建议优先使用 `Stop` 或 `StopVec`，以保证动作平稳结束。
- 在紧急情况下，应使用 `Abort` 确保设备安全快速停止。
- 使用任何停止方法后，应结合状态监控确保运动已完全终止，避免损伤。

9 控制 PCsuite IDE+

9.1 程序 API

ProgRun（运行线程任务）

	bool ProgRun(MotionController controller, AxisRef axis, int threadNumber, int taskNumber)	
描述	运行指定线程序列任务	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	threadNumber	线程号
	taskNumber	任务号
返回值	发送成功返回 true，发送失败返回 false	
备注	在指定线程中运行任务；任务为 taskNumber，线程为 threadNumber	

AutoExec（上电自动运动 IDE+）

	bool AutoExec(MotionController controller, AxisRef axis, bool enable)	
描述	自动执行任务	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	enable	是否启用自动执行
返回值	发送成功返回 true，发送失败返回 false	
备注	设置该轴在重启后是否自动执行预定义程序	

ProgReset（重置程序）

	bool ProgReset(MotionController controller, AxisRef axis, int threadNumber)	
描述	重置程序	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	threadNumber	线程号
返回值	发送成功返回 true，发送失败返回 false	
备注	重置对应线程序列任务，清除暂停或异常状态	

ProgHalt（暂停线程）

	bool ProgHalt(MotionController controller, AxisRef axis, int threadNumber)
描述	暂停指定线程的程序

参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	threadNumber	线程号
返回值	发送成功返回 true，发送失败返回 false	
备注	暂停当前线程	

ProgHaltAll（暂停所以线程）

	bool ProgHaltAll(MotionController controller, AxisRef axis, int threadNumber)
描述	停止所有线程
参数	MotionController controller 控制器实例
	AxisRef axisRef 轴的标识
	threadNumber 线程号
返回值	发送成功返回 true，发送失败返回 false
备注	停止所有线程执行的程序，可配合 ProgRun 继续执行

9.2 关键字说明

关键字	作用说明	举例
ProgRun	在指定线程中运行任务，任务号为 taskNumber，线程为 threadNumber	AProgRun[3],5 — 在线程 3 启动任务 5
1	代表“主程序”或“主任务”，可在 taskNumber 中设置，启动主程序	AProgRun[3],1 — 运行主程序
AutoExec=1	设置程序在上电或软件重启后自动执行，需保存到闪存	先设置后调用保存命令确保设置有效
ProgReset	重新初始化程序，清除暂停状态和异常，回到程序起点	在修改程序后，建议调用以确保清除异常
ProgHalt	暂停当前线程中的程序，暂停点可续传	AProgHalt[3] — 暂停线程 3
ProgHaltAll	停止所有程序运行，适用于清理系统状态或调试	在测试中确保所有程序暂停
ProgHaltAllLogHalt	停止所有程序且记录日志，便于排查问题	用于自动化故障分析

9.3 API 使用示例

4.1 运行任务

```
// 在线程 3 中运行编号为 5 的子程序
AAMotionAPI.ProgRun(controller, AxisRef.A, 3, 5);
```

说明：此命令将以线程 3 启动编号为 5 的任务程序，适用于特定任务调度。

4.2 运行主程序

```
// 在线程 3 中启动主程序（使用任务号-1 表示主程序）
AAMotionAPI.ProgRun(controller, AxisRef.A, 3, -1);
```

说明：在指定线程（3）中启动主程序（通常是整体控制流程）。

4.3 设定自动执行

```
// 设置当前轴自动执行，下次上电或重启后自动运行程序
AAMotionAPI.AutoExec(controller, AxisRef.A, true);
```

操作：启用自动执行功能。设置完成后，为保证效果生效，建议调用保存命令将设置存入闪存（根据设备支持的存储命令实现）。

4.4 任务重置

```
// 重置线程 3 的程序状态，恢复到初始状态
AAMotionAPI.ProgReset(controller, AxisRef.A, 3);
```

说明：重置线程 3 中的程序状态，适合在程序异常或需要重新开始时使用。

4.5 暂停线程

```
// 暂停线程 3 当前的程序
AAMotionAPI.ProgHalt(controller, AxisRef.A, 3);
```

说明：暂停指定线程中的程序，暂停点为当前位置。再次调用 `ProgRun` 可以从暂停点继续执行。

4.6 停止所有程序

```
// 停止控制器上所有线程中的程序
AAMotionAPI.ProgHaltAll(controller, AxisRef.A, threadNumber);
```

说明：停止所有正在运行的程序，用于安全检测或者系统调试。

9.4 说明

- **确保保存**：设置 `AutoExec=1` 后，务必用保存命令保存到闪存，防止掉电丢失。
- **任务优先级**：合理安排 `threadNumber` 和 `taskNumber`，避免冲突或等待。
- **暂停与重启**：`ProgHalt` 暂停后可用 `ProgRun` 继续，不影响程序状态。
- **异常复位**：长时间卡死请调用 `ProgReset` 重置。

10 CNC 运动

本节介绍通过上位机 API 实现对运动组（通常为多个轴的同步控制）的操作，包括启动、暂停、停止、轨迹运动等。

10.1 CNC-API

Begin（开始 CNC 运动）

	Bool Begin(MotionController controller)	
描述	开始 CNC 运动	
参数	MotionController controller	控制器实例
返回值	发送成功返回 true，发送失败返回 false	
备注	如成功，CNC 运动开始	

Pause（暂停 CNC 运动）

	bool Pause(MotionController controller)	
描述	暂停 CNC 运动	
参数	MotionController controller	控制器实例
返回值	发送成功返回 true，发送失败返回 false	
备注	暂停当前运动，状态可恢复	

Resume（恢复 CNC 运动）

	bool Resume(MotionController controller)	
描述	恢复 CNC 运动	
参数	MotionController controller	控制器实例
返回值	发送成功返回 true，发送失败返回 false	
备注	继续暂停的组运动	

ClearBuffer（清除 CNC 缓存区）

	bool ClearBuffer(MotionController controller)	
描述	清除缓冲区	
参数	MotionController controller	控制器实例
返回值	发送成功返回 true，发送失败返回 false	
备注	清除已排队的运动命令	

GroupStop（停止 CNC 运动）

	bool GroupStop(MotionController controller)	
描述	停止组运动	
参数	MotionController controller	控制器实例
返回值	发送成功返回 true，发送失败返回 false	
备注	立即停止组运动，可能未完成当前轨迹	

LinearAbsolute（CNC 线性绝对运动）

	bool LinearAbsolute(MotionController controller, int? aPos, int? bPos, int? cPos, int velCruise, int velEnd)	
描述	CNC 线性绝对运动（ABC 轴）	
参数	MotionController controller	控制器实例
	int? aPos	CNC 运动 A 轴位置（可选）
	int? aPos	CNC 运动 A 轴位置（可选）
	int? aPos	CNC 运动 A 轴位置（可选）
	int velCruise	CNC 运动运动速度
	int velEnd	CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false	
备注	执行同步线性段，最多可同时用于 3 个轴	

LinearAbsoluteX（X 轴线性运动）

	Bool LinearAbsoluteX(MotionController controller, int xPos, int velCruise, int velEnd)	
描述	X 轴线性运动	
参数	MotionController controller	控制器实例
	int xPos	CNC 运动 X 轴位置
	int velCruise	CNC 巡航速度
	int velEnd	CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false	
备注	只控制 X 轴的线性运动	

LinearAbsoluteY（Y 轴线性运动）

	Bool LinearAbsoluteY(MotionController controller, int yPos, int velCruise, int velEnd)	
描述	Y 轴线性运动	
参数	MotionController controller	控制器实例
	int yPos	CNC 运动 Y 轴位置
	int velCruise	CNC 巡航速度

	int velEnd	CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false	
备注	只控制 Y 轴的线性运动	

LinearAbsoluteZ（Z 轴线性运动）

	Bool LinearAbsoluteZ(MotionController controller, int yPos, int velCruise, int velEnd)
描述	Z 轴线性运动
参数	MotionController controller 控制器实例
	int zPos CNC 运动 Z 轴位置
	int velCruise CNC 巡航速度
	int velEnd CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false
备注	只控制 Z 轴的线性运动

LinearAbsoluteXY（XY 轴线性运动）

	Bool LinearAbsoluteXY(MotionController controller, int xPos, int yPos, int velCruise, int velEnd)
描述	XY 轴线性运动
参数	MotionController controller 控制器实例
	int xPos CNC 运动 X 轴位置
	int yPos CNC 运动 Y 轴位置
	int velCruise CNC 巡航速度
	int velEnd CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false
备注	执行同步线性段，控制 XY 轴的运动

LinearAbsoluteXZ（XZ 轴线性运动）

	Bool LinearAbsoluteXZ(MotionController controller, int xPos, int zPos, int velCruise, int velEnd)
描述	XZ 轴线性运动
参数	MotionController controller 控制器实例
	int xPos CNC 运动 X 轴位置
	int zPos CNC 运动 Z 轴位置
	int velCruise CNC 巡航速度
	int velEnd CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false

备注	执行同步线性段，控制 XZ 轴的运动	

LinearAbsoluteYZ（YZ 轴线性运动）

	Bool LinearAbsoluteYZ(MotionController controller, int yPos, int zPos, int velCruise, int velEnd)	
描述	YZ 轴线性运动	
参数	MotionController controller	控制器实例
	int yPos	CNC 运动 Y 轴位置
	int zPos	CNC 运动 Z 轴位置
	int velCruise	CNC 巡航速度
	int velEnd	CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false	
备注	执行同步线性段，控制 yZ 轴的运动	

LinearAbsoluteXYZ（XYZ 轴线性运动）

	Bool LinearAbsoluteXYZ(MotionController controller, int xPos, int yPos, int zPos, int velCruise, int velEnd)	
描述	XYZ 轴线性运动	
参数	MotionController controller	控制器实例
	int xPos	CNC 运动 X 轴位置
	int yPos	CNC 运动 Y 轴位置
	int zPos	CNC 运动 Z 轴位置
	int velCruise	CNC 巡航速度
	int velEnd	CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false	
备注	执行同步线性段，控制 XYZ 轴的运动	

ArcXY（XY 轴弧性运动）

	Bool ArcXY(MotionController controller, int xPos, int yPos, int iPos, int jPos, int dir, int velCruise, int velEnd)	
描述	XY 弧形运动	
参数	MotionController controller	控制器实例
	int xPos	CNC 运动 X 轴终点坐标
	int yPos	CNC 运动 Y 轴终点坐标
	int iPos	CNC 运动圆心位置
	int jPos	CNC 运动圆心位置

	int dir	方向（1-顺时针，0-逆时针）
	int velCruise	巡航速度
	int velEnd	CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false	
备注	执行同步的弧线运动，XY 弧形运动	

	Bool ArcXY(MotionController controller, int xPos, int yPos, int iPos, int jPos, int dir, int velCruise, int velEnd， int？ additionalCycles)

描述	XY 弧线运动	
参数	MotionController controller	控制器实例
	int xPos	CNC 运动 X 轴终点坐标
	int yPos	CNC 运动 Y 轴终点坐标
	int iPos	CNC 运动圆心位置
	int jPos	CNC 运动圆心位置
	int dir	方向（1-顺时针，0-逆时针）
	int velCruise	巡航速度
	int velEnd	CNC 运动结束速度
	Int？ additionalCycles	附加圈数（可选）
返回值	发送成功返回 true，发送失败返回 false	
备注	执行同步的弧线运动，支持多圈运动	

ArcXZ（XZ 轴弧性运动）

	Bool ArcXZ(MotionController controller, int xPos, int zPos, int iPos, int jPos, int dir, int velCruise, int velEnd， int？ additionalCycles)

描述	XZ 弧线运动	
参数	MotionController controller	控制器实例
	int xPos	CNC 运动 X 轴终点坐标
	int zPos	CNC 运动 Z 轴终点坐标
	int iPos	CNC 运动圆心位置
	int jPos	CNC 运动圆心位置
	int dir	方向（1-顺时针，0-逆时针）
	int velCruise	巡航速度
	int velEnd	CNC 运动结束速度
	Int？additionalCycles	附加圈数（可选）
返回值	发送成功返回 true，发送失败返回 false	

备注	执行同步的弧线运动，支持多圈运动

ArcYZ（YZ 轴弧性运动）

	Bool ArcYZ(MotionController controller, int yPos, int zPos, int iPos, int jPos, int dir, int velCruise, int velEnd， int？ additionalCycles)	
描述	YZ 弧线运动	
参数	MotionController controller	控制器实例
	int yPos	CNC 运动 Y 轴终点坐标
	int zPos	CNC 运动 Z 轴终点坐标
	int iPos	CNC 运动圆心位置
	int jPos	CNC 运动圆心位置
	int dir	方向（1-顺时针，0-逆时针）
	int velCruise	巡航速度
	int velEnd	CNC 运动结束速度
	Int？ additionalCycles	附加圈数（可选）
返回值	发送成功返回 true，发送失败返回 false	
备注	执行同步的弧线运动，支持多圈运动	

Delay（CNC 运动延时）

	Bool Delay(MotionController controller, int ms)	
描述	延时等待	
参数	MotionController controller	控制器实例
	int ms	延时时间（毫秒）
返回值	发送成功返回 true，发送失败返回 false	
备注	在段之间设置延迟	

SetMotionProfile（设置运动矢量参数）

	Bool SetMotionProfile(MotionController controller, int percentage, int accel, int decel, double sfactor)	
描述	设置运动矢量参数	
参数	MotionController controller	控制器实例
	int percentage	速度百分比
	int accel	加速度
	int decel	减速度
	double sfactor	平滑系数
返回值	发送成功返回 true，发送失败返回 false	

备注	从该点设置向量运动参数可以在 CNC 运动中多次使用

SetCornerParams（设置拐角参数）

	Bool SetCornerParams(MotionController controller, int cornerType, int radiusMethod, int radiusOrError, int axisAccel, int minAngle, int accelLimitType)	
描述	设置拐角参数	
参数	MotionController controller	控制器实例
	int cornerType	角类型
	int radiusMethod	拐角半径方式
	int radiusOrError	拐角半径
	int axisAccel	拐角轴加速度
	int minAngle	拐角轴最小速度
	int accelLimitType	轴加速限制类型（0-不使用， 1-使用）
返回值	发送成功返回 true，发送失败返回 false	
备注	定义在所有后续自动拐角运动段中使用的一般拐角参数（类型、半径等）	

SetStartPositions（设置起始位置）

	Bool SetStartPositions(MotionController controller, AxisRef axisRef, int xPos, int yPos, int zPos)	
描述	设置起始位置	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴引用
	int xPos	设置 CNC 运动的 X 开始位置
	int yPos	设置 CNC 运动的 Y 开始位置
	int zPos	设置 CNC 运动的 Z 开始位置
返回值	发送成功返回 true，发送失败返回 false	
备注	设置所有成员轴的预期初始位置。每台 CNC 运动一次,放到第一个段	

WriteDOutPort（写入离散输出）

	Bool WriteDOutPort(MotionController controller, int dOutPortValue)	
描述	写入离散输出	
参数	MotionController controller	控制器实例
	int dOutPortValue	输出值（十进制）
返回值	发送成功返回 true，发送失败返回 false	
备注	将指定值写入 DOutPort 到指定端口（十进制）	

GroupDOutSetBit（设置离散输出）

	Bool GroupDOutSetBit(MotionController controller, AxisRef axisRef, int bitMask)	
描述	设置离散输出（置 1）	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴引用（默认写 A）
	int bitMask	位掩码
返回值	发送成功返回 true，发送失败返回 false	
备注	设置指定比特位	

GroupDOutClearBit（清除离散输出）

	Bool GroupDOutClearBit(MotionController controller, AxisRef axisRef, int bitMask)	
描述	清除离散输出（置 0）	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴引用（默认写 A）
	int bitMask	位掩码
返回值	发送成功返回 true，发送失败返回 false	
备注	清除指定比特位	

GroupDOutToggleBit（翻转离散输出）

	Bool GroupDOutToggleBit(MotionController controller, AxisRef axisRef, int bitMask)	
描述	翻转离散输出	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴引用（默认写 A）
	int bitMask	位掩码
返回值	发送成功返回 true，发送失败返回 false	
备注	翻转指定比特位	

WriteGenData（写 GenData[]数组数据）

	Bool WriteGenData(MotionController controller, int gendataIndex, int gendataValue)	
描述	写 GenData[]数组数据	
参数	MotionController controller	控制器实例
	int gendataIndex	GenData 索引
	int gendataValue	GenData 索引所对应的值
返回值	发送成功返回 true，发送失败返回 false	
备注	将指定值写入 GenData[] 数组的指定索引	

WriteUserParam（写 UserParam[] 数组数据）

	Bool WriteUserParam(MotionController controller, AxisRef axisRef, int userparamIndex, int userparamValue)	
描述	写 UserParam[] 数组数据	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴索引
	int userparamIndex	UserParam 索引
	int userparamValue	UserParam 索引所对应的值
返回值	发送成功返回 true，发送失败返回 false	
备注	将指定值写入 UserParam[] 数组的指定索引	

GenDataWait（使用用户数组段等待）

	Bool GenDataWait(MotionController controller, int gendataIndex, int trigType, int trigVal)	
描述	使用用户数组段等待	
参数	MotionController controller	控制器实例
	int gendataIndex	使用的数组索引
	int trigType	触发器类型
	int trigVal	触发器值
返回值	发送成功返回 true，发送失败返回 false	
备注	当到达这种类型的段时，CNC 引擎将等待，直到所有成员轴处于满足段参数定义的触发条件为止	

UserParamWait（使用用户数组段等待）

	Bool UserParamWait(MotionController controller, AxisRef axis, int userparamIndex, int trigType, int trigVal)	
描述	使用用户数组段等待	
参数	MotionController controller	控制器实例
	AxisRef axis	轴索引
	int userparamIndex	userparam 索引
	int trigType	触发器类型
	int trigVal	触发器值
返回值	发送成功返回 true，发送失败返回 false	
备注	当到达这种类型的段时，CNC 引擎将等待，直到所有成员轴处于满足段参数定义的触发条件为止	

SetCurrPositions（设置位置）

	Bool SetCurrPositions(MotionController controller, AxisRef axis, int? aPos, int? bPos, int? cPos)	
描述	设置（分配）位置	
参数	MotionController controller	控制器实例
	AxisRef axis	轴索引
	int? aPos	设置 X 轴位置（可选）
	int? bPos	设置 Y 轴位置（可选）
	int? cPos	设置 Z 轴位置（可选）
返回值	发送成功返回 true，发送失败返回 false	
备注	为选定的成员轴分配新的位置值（必须不在移动中）	

AutoCorner（自动拐角运动）

	Bool AutoCorner(MotionController controller, AxisRef axis)	
描述	自动拐角运动	
参数	MotionController controller	控制器实例
	AxisRef axis	轴索引
返回值	发送成功返回 true，发送失败返回 false	
备注	根据最后设置的拐角参数段请求自动计算拐角	

SetMaxVelJumpParams（设置最大速度跳跃参数）

	Bool SetMaxVelJumpParams(MotionController controller, int? aMaxVJ, int? bMaxVJ, int? cMaxVJ, int jumpMode)	
描述	设置最大速度跳跃参数	
参数	MotionController controller	控制器实例
	int? aMaxVJ	A 轴最大跳跃速度（可空）
	int? bMaxVJ	B 轴最大跳跃速度（可空）
	int? cMaxVJ	C 轴最大跳跃速度（可空）
	int jumpMode	速度跳跃模式 0-忽略 1-使用
返回值	发送成功返回 true，发送失败返回 false	
备注	定义每个轴的最大速度跳跃,其中线性运动段后面跟着另一个线性运动段(没有自动拐角运动段）	

SetMaxAccelParams（设置轴最大加速度）

	Bool SetMaxAccelParams(MotionController controller, int? aMaxAccel, int? bMaxAccel, int? cMaxAccel)
描述	设置轴最大加速度

参数	MotionController controller	控制器实例
	int? aMaxAccel	A 轴最大加速度（可空）
	int? bMaxAccel	B 轴最大加速度（可空）
	int? cMaxAccel	C 轴最大加速度（可空）
返回值	发送成功返回 true，发送失败返回 false	
备注	仅当在拐角参数段中后用时使用	

MultiWriteGenData（多次写入 GenData）

	Bool MultiWriteGenData(MotionController controller, int gendataIndex, int gendataValue1, int gendataValue2, int gendataValue3, int gendataValue4)	
描述	多次写入用户数组	
参数	MotionController controller	控制器实例
	int gendataIndex	GenData[]数据索引
	int gendataValue1	GenData[x]数据值
	int gendataValue12	GenData[X +1]数据值
	int gendataValue13	GenData[X +2]数据值
	int gendataValue14	GenData[X +3]数据值
返回值	发送成功返回 true，发送失败返回 false	
备注	将 4 个值写入 GenData[] 数组的 4 个连续索引	

MultiWriteUserParam（多次写入 UserParam）

	Bool MultiWriteUserParam(MotionController controller, AxisRef axis, int userparamIndex, int userparamValue1, int userparamValue2, int userparamValue3, int userparamValue4)	
描述	多次写入用户数组	
参数	MotionController controller	控制器实例
	AxisRef axis	轴索引
	int userparamIndex	UserParam[]数据索引
	int userparamValue1	UserParam[x]数据值
	int userparamValue2	UserParam[X +1]数据值
	int userparamValue3	UserParam[X +2]数据值
	int userparamValue4	UserParam[X +3]数据值
返回值	发送成功返回 true，发送失败返回 false	
备注	将 4 个值写入 UserParam[] 数组的 4 个连续索引	

MultiWriteGenDataWait（多次写入 GenData 并等待）

	Bool MultiWriteGenDataWait(MotionController controller, int gendataIndex, int gendataValue1, int gendataValue2, int gendataValue3, int gendataValue4, int trigIndex, int trigTyp, int trigVal)	
描述	多次写入用户数组并等待	
参数	MotionController controller	控制器实例
	int gendataIndex	GenData[]数据索引
	int gendataValue1	GenData[x]数据值
	int gendataValue12	GenData[X +1]数据值
	int gendataValue13	GenData[X +2]数据值
	int gendataValue14	GenData[X +3]数据值
	int trigIndex	触发条件索引
	int trigType	触发器类型
	int trigVal	触发器值
返回值	发送成功返回 true，发送失败返回 false	
备注	将 4 个值写入 GenData[]或 UserParam[]数组的 4 个连续索引,然后在所有成员轴的位置等待，直到满足段参数定义的触发条件	

MultiWriteUserParamWait

	Bool MultiWriteUserParamWait(MotionController controller, AxisRef axis, int userparamIndex, int userparamValue1, int userparamValue2, int userparamValue3, int userparamValue4, int trigIndex, int trigTyp, int trigVal)	
描述	多次写入用户数组并等待	
参数	MotionController controller	控制器实例
	AxisRef axis	轴索引
	int userparamIndex	UserParam[]数据索引
	int userparamValue1	UserParam[x]数据值
	int userparamValue2	UserParam[X +1]数据值
	int userparamValue3	UserParam[X +2]数据值
	int userparamValue4	UserParam[X +3]数据值
	int trigIndex	触发条件索引
	int trigType	触发器类型
	int trigVal	触发器值
返回值	发送成功返回 true，发送失败返回 false	
备注	将 4 个值写入 UserParam[]数组的 4 个连续索引,然后在所有成员轴的位置等待，直到满足段参数定义的触发条件	

10.2 CNC-API（CiGroup）

LinearAbsolute（Ci-CNC 线性绝对运动）

	Bool LinearAbsolute(MotionController, AxisRef, int? aPos, int? bPos, int? cPos, int? dPos, int velCruise, int velEnd)	
描述	CNC 线性绝对运动（ABCD 轴）	
参数	MotionController controller	控制器实例
	AxisRef axis	轴引用，当是 AxisRef.A 选择的是 GroupA。当是 AxisRef.B 选择的是 GroupB
	int? aPos	CNC 运动 A 轴位置（可选）
	int? bPos	CNC 运动 B 轴位置（可选）
	int? cPos	CNC 运动 C 轴位置（可选）
	int? dPos	CNC 运动 D 轴位置（可选）
	int velCruise	CNC 运动运动速度
	int velEnd	CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false	
备注	执行同步线性段，最多可同时用于 4 个轴	

	Bool LinearAbsolute(MotionController, int? aPos, int? bPos, int? cPos, int? dPos, int? ePos, int? fPos, int velCruise, int velEnd)	
描述	CNC 线性绝对运动（ABCD 轴）	
参数	MotionController controller	控制器实例
	int? aPos	CNC 运动 A 轴位置（可选）
	int? bPos	CNC 运动 B 轴位置（可选）
	int? cPos	CNC 运动 C 轴位置（可选）
	int? dPos	CNC 运动 D 轴位置（可选）
	int? ePos	CNC 运动 E 轴位置（可选）
	int? fPos	CNC 运动 F 轴位置（可选）
	int velCruise	CNC 运动运动速度
	int velEnd	CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false	
备注	执行同步线性段，最多可同时用于 6 个轴	

Bool LinearAbsolute(MotionController, int? aPos, int? bPos, int? cPos, int? dPos, int? ePos, int? fPos, int velCruise, int velEnd)

描述	CNC 线性绝对运动（ABCDEF 轴）	
参数	MotionController controller	控制器实例
	AxisRef axis	轴引用，当是 AxisRef.A 选择的是 GroupA。当是 AxisRef.B 选择的是 GroupB
	int? aPos	CNC 运动 A 轴位置（可选）
	int? bPos	CNC 运动 B 轴位置（可选）
	int? cPos	CNC 运动 C 轴位置（可选）
	int? dPos	CNC 运动 D 轴位置（可选）
	int? ePos	CNC 运动 E 轴位置（可选）
	int? fPos	CNC 运动 F 轴位置（可选）
	int velCruise	CNC 运动运动速度
	int velEnd	CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false	
备注	执行同步线性段，最多可同时用于 6 个轴	

LinearAbsoluteT（T 轴线性绝对运动）

	Bool LinearAbsoluteT(MotionController, AxisRef, int tPos, int velCruise, int velEnd)
描述	CNC 线性绝对运动（T 轴）
参数	MotionController controller 控制器实例
	AxisRef axis 轴引用，当是 AxisRef.A 选择的是 GroupA。当是 AxisRef.B 选择的是 GroupB
	int tPos CNC 运动 T 轴位置
	int velCruise CNC 运动运动速度
	int velEnd CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false
备注	执行同步线性段，最多可同时用于 1 个轴

LinearAbsoluteXT（XT 轴线性绝对运动）

	Bool LinearAbsoluteXT(MotionController, AxisRef, int xPos, int tPos, int velCruise, int velEnd)
描述	CNC 线性绝对运动（XT 轴）
参数	MotionController controller 控制器实例
	AxisRef axis 轴引用，当是 AxisRef.A 选择的是 GroupA。当是 AxisRef.B 选择的是 GroupB

	int xPos	CNC 运动 X 轴位置
	int tPos	CNC 运动 T 轴位置
	int velCruise	CNC 运动运动速度
	int velEnd	CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false	
备注	执行同步线性段，最多可同时用于 2 个轴	

LinearAbsoluteYT（YT 轴线性绝对运动）

	Bool LinearAbsoluteYT(MotionController, AxisRef, int yPos, int tPos, int velCruise, int velEnd)
描述	CNC 线性绝对运动（YT 轴）
参数	MotionController controller　控制器实例
	AxisRef axis　轴引用，当是 AxisRef.A 选择的是 GroupA。当是 AxisRef.B 选择的是 GroupB
	int yPos　CNC 运动 Y 轴位置
	int tPos　CNC 运动 T 轴位置
	int velCruise　CNC 运动运动速度
	int velEnd　CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false
备注	执行同步线性段，最多可同时用于 2 个轴

LinearAbsoluteZT（ZT 轴线性绝对运动）

	Bool LinearAbsoluteZT(MotionController, AxisRef, int zPos, int tPos, int velCruise, int velEnd)
描述	CNC 线性绝对运动（ZT 轴）
参数	MotionController controller　控制器实例
	AxisRef axis　轴引用，当是 AxisRef.A 选择的是 GroupA。当是 AxisRef.B 选择的是 GroupB
	int zPos　CNC 运动 Z 轴位置
	int tPos　CNC 运动 T 轴位置
	int velCruise　CNC 运动运动速度
	int velEnd　CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false
备注	执行同步线性段，最多可同时用于 2 个轴

LinearAbsoluteXYT（XYT 轴线性绝对运动）

	Bool LinearAbsoluteXYT(MotionController, AxisRef, int xPos, int yPos, int tPos, int velCruise, int velEnd)	
描述	CNC 线性绝对运动（XYT 轴）	
参数	MotionController controller	控制器实例
	AxisRef axis	轴引用，当是 AxisRef.A 选择的是 GroupA。当是 AxisRef.B 选择的是 GroupB
	int xPos	CNC 运动 X 轴位置
	int yPos	CNC 运动 Y 轴位置
	int tPos	CNC 运动 T 轴位置
	int velCruise	CNC 运动运动速度
	int velEnd	CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false	
备注	执行同步线性段，最多可同时用于 3 个轴	

LinearAbsoluteXZT（XZT 轴线性绝对运动）

	Bool LinearAbsoluteXZT(MotionController, AxisRef, int xPos, int zPos, int tPos, int velCruise, int velEnd)	
描述	CNC 线性绝对运动（XZT 轴）	
参数	MotionController controller	控制器实例
	AxisRef axis	轴引用，当是 AxisRef.A 选择的是 GroupA。当是 AxisRef.B 选择的是 GroupB
	int xPos	CNC 运动 X 轴位置
	int zPos	CNC 运动 Z 轴位置
	int tPos	CNC 运动 T 轴位置
	int velCruise	CNC 运动运动速度
	int velEnd	CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false	
备注	执行同步线性段，最多可同时用于 3 个轴	

LinearAbsoluteYZT（YZT 轴线性绝对运动）

	Bool LinearAbsoluteYZT(MotionController, AxisRef, int yPos, int zPos, int tPos, int velCruise, int velEnd)	
描述	CNC 线性绝对运动（YZT 轴）	
参数	MotionController controller	控制器实例

	AxisRef axis	轴引用，当是 AxisRef.A 选择的是 GroupA。当是 AxisRef.B 选择的是 GroupB
	int yPos	CNC 运动 Y 轴位置
	int zPos	CNC 运动 Z 轴位置
	int tPos	CNC 运动 T 轴位置
	int velCruise	CNC 运动运动速度
	int velEnd	CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false	
备注	执行同步线性段，最多可同时用于 3 个轴	

LinearAbsoluteXYZT（XYZT 轴线性绝对运动）

	Bool LinearAbsoluteXYZT(MotionController, AxisRef, int xPos, int yPos, int zPos, int tPos, int velCruise, int velEnd)
描述	CNC 线性绝对运动（XYZT 轴）

参数	MotionController controller	控制器实例
	AxisRef axis	轴引用，当是 AxisRef.A 选择的是 GroupA。当是 AxisRef.B 选择的是 GroupB
	int xPos	CNC 运动 X 轴位置
	int yPos	CNC 运动 Y 轴位置
	int zPos	CNC 运动 Z 轴位置
	int tPos	CNC 运动 T 轴位置
	int velCruise	CNC 运动运动速度
	int velEnd	CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false	
备注	执行同步线性段，最多可同时用于 4 个轴	

ArcXT（XT 弧形运动）

	Bool ArcXT(MotionController, AxisRef, int xPos, int tPos, int iPos, int lPos, int dir, int velCruise, int velEnd)
描述	XT 弧形运动

参数	MotionController controller	控制器实例
	AxisRef axis	轴引用，当是 AxisRef.A 选择的是 GroupA。当是 AxisRef.B 选择的是 GroupB
	int xPos	CNC 运动 X 轴终点坐标

	int tPos	CNC 运动 T 轴终点坐标
	int iPos	CNC 运动圆心位置
	int lPos	CNC 运动圆心位置
	int dir	方向（1-顺时针，0-逆时针）
	int velCruise	巡航速度
	int velEnd	CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false	
备注	执行同步的弧线运动，XT 弧形运动	

ArcYT（YT 弧形运动）

	Bool ArcYT(MotionController, AxisRef, int yPos, int tPos, int jPos, int lPos, int dir, int velCruise, int velEnd)
描述	YT 弧形运动

参数	MotionController controller	控制器实例
	AxisRef axis	轴引用，当是 AxisRef.A 选择的是 GroupA。当是 AxisRef.B 选择的是 GroupB
	int yPos	CNC 运动 Y 轴终点坐标
	int tPos	CNC 运动 T 轴终点坐标
	int iPos	CNC 运动圆心位置
	int lPos	CNC 运动圆心位置
	int dir	方向（1-顺时针，0-逆时针）
	int velCruise	巡航速度
	int velEnd	CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false	
备注	执行同步的弧线运动，YT 弧形运动	

ArcZT（ZT 弧形运动）

	Bool ArcZT(MotionController, AxisRef, int zPos, int tPos, int kPos, int lPos, int dir, int velCruise, int velEnd)
描述	ZT 弧形运动

参数	MotionController controller	控制器实例
	AxisRef axis	轴引用，当是 AxisRef.A 选择的是 GroupA。当是 AxisRef.B 选择的是 GroupB
	int ZPos	CNC 运动 Z 轴终点坐标
	int TPos	CNC 运动 T 轴终点坐标

	int iPos	CNC 运动圆心位置
	int lPos	CNC 运动圆心位置
	int dir	方向（1-顺时针，0-逆时针）
	int velCruise	巡航速度
	int velEnd	CNC 运动结束速度
返回值	发送成功返回 true，发送失败返回 false	
备注	执行同步的弧线运动，ZT 弧形运动	

WriteICiGroupDOutPort（写入离散输出）

	Bool WriteICiGroupDOutPort(MotionController, AxisRef, int dOutPortValue)
描述	写入离散输出
参数	MotionController controller　控制器实例
	int dOutPortValue　输出值（十进制）
返回值	发送成功返回 true，发送失败返回 false
备注	将指定值写入 DOutPort 到指定端口（十进制）

10.3 CNC-API（AGM800-CiGroup） 示例

```
// 1. 初始化控制器实例
MotionController controller =
AAMotionAPI.Initialize(ControllerType.AGM800);

// 2. 设置每个轴的运动模式为 CNC（11 为 CNC 模式）
controller.GetAxis(AxisRef.A).MotionMode = 11;
controller.GetAxis(AxisRef.B).MotionMode = 11;
controller.GetAxis(AxisRef.C).MotionMode = 11;
controller.GetAxis(AxisRef.D).MotionMode = 11;

// 3. 清空运动缓冲区，准备下一次运动命令
AAMotionAPI.ClearBuffer(controller);

// 4. 下发第一组运动指令（A、B、C 轴同步运动到目标位置）
AAMotionAPI.LinearAbsolute(controller, AxisRef.A, 10000, 10000, 10000,
null, 10000, 0);
// 说明：
// - A：运动到位置 10000
// - B：运动到位置 10000
// - C：运动到位置 10000
// - 速度：10000
// - 结束速度：0

// 5. 下发第二组运动指令（B 轴运动到目标位置）
// 注：参数配置应符合 API 定义，此处示范假设你要运动 B 轴到位置 8888，第二个参数
“AxisRef.A”是使用 GroupA，对于 AGM800 支持使用 AxisRef.B 去使用 GroupB。
AAMotionAPI.LinearAbsolute(controller, AxisRef.A, null, 8888, null, null,
10000, 10000, 10000, 0);
// - 其他轴保持不变或未设置目标（null）
```

```
// 6. 开始运动，执行命令
AAMotionAPI.Begin(controller);

// 7. 等待 A 轴运动完成
while (controller.GetAxis(AxisRef.A).InTargetStat != 4) // 状态 4：已到位
{
    Thread.Sleep(10); //每 10ms 检查一次
}

// 8. 等待 B 轴运动完成
while (controller.GetAxis(AxisRef.B).InTargetStat != 4)
{
    Thread.Sleep(10);
}
```

10.4 CNC-API 示例

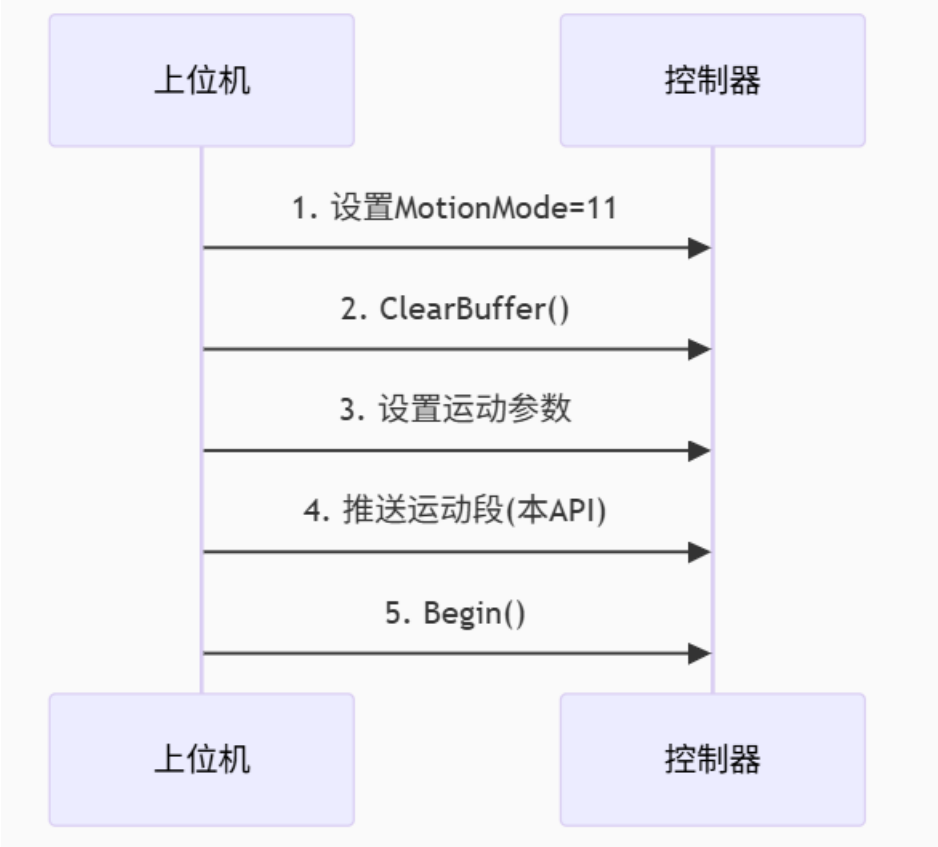
简介

本示例演示如何设置控制器为 CNC 运动模式，清除缓冲区，配置运动参数、起始位置及运动路径，最后启动运动。示例中涉及 X 轴、Y 轴运动以及 XY 组合运动，展示 CNC 运动的基本流程。

示例

```
// 1. 设置轴运动模式为 CNC 模式(值 11)
controller.GetAxis(AxisRef.A).MotionMode = 11;
controller.GetAxis(AxisRef.B).MotionMode = 11;

// 2. 清除运动缓冲区
AAMotionAPI.ClearBuffer(controller);

// 3. 设置运动配置参数：速度比例 100%，最大加速度/减速度 200000，平滑系数 0
AAMotionAPI.SetMotionProfile(controller, 100, 200000, 200000, 0);

// 4. 设置起始位置，X、Y 轴起点为 0，C 轴忽略（null）
AAMotionAPI.SetStartPositions(controller, 0, 0, null);

// 5. 运动命令示例
AAMotionAPI.LinearAbsoluteX(controller, 500, 50000, 10000);  // X 轴运动到
500，起始速度 50000，结束速度 10000
AAMotionAPI.LinearAbsoluteY(controller, 800, 50000, 50000);  // Y 轴运动到
800，起始速度 50000，结束速度 50000
AAMotionAPI.LinearAbsoluteXY(controller, 1000, 1000, 50000, 50000);  // XY
轴运动到(1000,1000)，起始速度 50000，结束速度 50000

AAMotionAPI.Delay(controller, 1000);  // 延时 1000ms 等待运动完成
// 6. 开始运动执行
AAMotionAPI.Begin(controller);

// 7.终点运动，速度结束设为 0，表示停止
AAMotionAPI.LinearAbsoluteXY(controller, 0, 0, 50000, 0);

// 8. 开始运动执行
```

```
AAMotionAPI.Begin(controller);
```



关键步骤说明

设置运动模式：
调用 `MotionMode = 11` 设置对应轴为 CNC 运动模式。控制器将在后续调用 `Begin` 时，根据此模式执行相应运动类型。
注意： 当前运动结束之前，不能更改 `MotionMode`。

缓冲区清理：
调用 `ClearBuffer` 确保无残留运动指令，避免干扰当前运动。

运动参数设置：
调用 `SetMotionProfile` 配置速度比例、最大加速减速及平滑系数，影响运动动态响应。

分段轨迹规划：
使用 `LinearAbsoluteX/Y/XY` 等 API 逐段写入运动路径，末段速度需设为 0 保证运动平滑停止。

启动执行：
调用 `Begin` 启动以上所有配置的运动指令。

10.5 ArcXY 带附加圈数的注意事项

在使用 AAMotionAPI.ArcXY 进行带附加圈数的圆弧运动时，必须特别注意以下关键点，否则会导致运动报错或者延时失效。

1. 结束速度必须设置为 0

API 示例调用：

```
controller.ErrorOccurred += (errorCode, msgSent, errorMsg) =>
{
    Console.WriteLine($"Message Sent: {msgSent}");
    Console.WriteLine($"Error Code: {errorCode}");
    Console.WriteLine(errorMsg);

};
AAMotionAPI.ArcXY(controller, 100, 100, 500, 500, 1, 10000, 0, 3);
```

- 说明：
  - 终点坐标为 (100, 100)
  - 圆心坐标为 (500, 500)
  - 转动方向：顺时针（dir=1）
  - 起始速度为 10000
  - **结束速度必须为 0**
  - 附加圈数为 3，实际运动圈数为 1 + 3 = 4 圈

必选参数注意：
在使用附加参数时，结束速度不可为非零，否则会报错：

```
Error:204
ERR 204
Sent: 83886079,2000
Error: The last motion segment has a non zero end speed and this segment
can't continue it (not the same involved axes or a motion-blocking segment)
```

2. 起点、圆心、终点必须构成有效圆弧

确保起始点、圆心坐标与终点能够组成一个正确的圆弧，否则指令将执行失败。

3. 延时操作注意

如果在弧线运动后面接延时指令，且未将结束速度设为 0，则延时命令不会生效。

4.总结

- **结束速度必须为 0**，否则报错且后续延时失效
- 圆弧参数（起点、圆心、终点）必须符合圆形几何关系
- 附加圈数参数用来实现多圈运动，实际绕圈数=1 + 附加圈数

10.6 MotionMode 关键字说明

值	运动类型	备注
0	Jog	Jog – 电机将达到 Speed 中设置的速度，并在接收到 Stop 命令之前以恒定速度继续。运动使用的加速和减速由 Accel 和 Decel 定义。运动也可以根据 Jerk 的值进行平滑。
1	PTP (点位到点位)	PTP – 移动到由 RelTrgt 或 AbsTrgt（如果 RelTrgt = 0）定义的位置。运动的其他参数由 Accel、Speed、Decel 和 Jerk 定义。
2	PTP 重复	PTP 重复 – 从当前位置到由 RelTrgt 或 AbsTrgt（如果 RelTrgt = 0）定义的位置的重复运动，然后返回。运动的其他参数由 Accel、Speed、Decel 和 Jerk 定义。 电机将在运动的每一端等待由 RptWait 确定的时间。运动使用 StopRep 停止。
3	脉冲方向直接模式	脉冲方向直接模式 – 位置环的参考位置直接从脉冲方向输入接收。
4	脉冲方向间接模式	脉冲方向间接模式 – 此运动的位置信号根据脉冲方向输入确定。然而，接收到的脉冲数不会立即用作位置环的参考。 接收到的脉冲数被添加到当前位置目标。分析器使用所有运动参数的值构建到新目标的运动轨迹：Accel、Decel、Speed、Jerk。 这意味着即使一次输入大量脉冲，结果运动也不会像阶跃响应，而是更平滑。请注意，对快速输入的响应将更平滑，但也更慢。
5	齿轮直接运动	齿轮直接运动
6	齿轮间接运动	
7	ECAM 直接运动	
8	ECAM 间接运动	
9	FIFO 运动	
10	主从位置参考	
11	CNC 组 A 运动	本示例使用，适用于多轴联动 CNC 控制
12	摇杆直接运动	摇杆直接运动（模拟信号位置参考）
13	摇杆间接运动	摇杆间接运动（模拟信号位置参考，用户可设置）
14	摇杆直接运动	摇杆直接运动，输入模拟信号表示速度参考
15	摇杆间接运动	摇杆间接运动，输入模拟信号表示速度参考，此分析器可由用户设

值	运动类型	备注
		置
16	矢量运动	
17	CNC 组 B 运动	AGM800 支持
18	样条缓冲运动	

11 数字 IO 输出

本节描述数字输出（DOutPort）相关 API 接口，支持针对指定轴 IO 模块的数字输出位的清除、设置及切换操作。接口底层通过通讯接口发送命令并等待响应。

11.1 数字 IO-API

DOutClearBit（清除数字输出位）

	bool DOutClearBit(MotionController controller, AxisRef axis, int index)	
描述	清除数字输出位	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int index	数字输出口位索引，从 0 开始
返回值	true：指令发送成功，操作执行完成。	
	false：指令发送失败或通信异常	
备注	将指定轴的数字输出口的第 index 位清零（复位）	

DOutSetBit（设置数字输出位）

	bool DOutSetBit(MotionController controller, AxisRef axis, int index)	
描述	设置数字输出位	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int index	数字输出口位索引，从 0 开始
返回值	true：指令发送成功，操作执行完成。	
	false：指令发送失败或通信异常	
备注	将指定轴的数字输出口的第 index 位置 1（置位）	

DOutToggleBit（翻转数字输出位）

	bool DOutToggleBit(MotionController controller, AxisRef axis, int index)	
描述	翻转数字输出位	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int index	数字输出口位索引，从 0 开始
返回值	true：指令发送成功，操作执行完成。	
	false：指令发送失败或通信异常	

备注	将指定轴的数字输出口的第 index 位清零（复位）

GetDOutPort（得到数字输出位）

	int GetDOutPort(MotionController controller, AxisRef axis , int bit)	
描述	得到数字输出位	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int bit	数字输出口位索引，从 0 开始
返回值	获取特定数字输出位的当前状态	
备注	指定位的状态（0 = Off, 1 = On）。还有必要参考 DOutLog	

GetDInPort（得到数字输入位）

	int GetDInPort(MotionController controller, AxisRef axis , int bit)	
描述	得到数字输入位	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int bit	数字输入口位索引，从 0 开始
返回值	获取特定数字输入位的当前状态	
备注	指定位的状态（0 = Off, 1 = On）。还有必要参考 DInLog	

11.2 数字 IO 示例

```
// 清除 A 轴数字输出口第 3 位
AAMotionAPI.DOutClearBit(controller, AxisRef.A, 3);

// 设置 B 轴数字输出口第 1 位
AAMotionAPI.DOutSetBit(controller, AxisRef.B, 1);

// 切换 C 轴数字输出口第 0 位
AAMotionAPI.DOutToggleBit(controller, AxisRef.A, 0);

// 获取 A 轴数字输入端口 2 的状态
AAMotionAPI.GetDInPort(controller, AxisRef.A, 2);

// 获取 A 轴数字输出端口 3 的状态
AAMotionAPI.GetDOutPort(controller, AxisRef.A, 3);
```

11.3 注意事项

- index 参数必须在硬件支持的范围内，避免越界。

- 操作成功返回 `true`，建议调用后检查返回值确保执行正确。
- 数字输出操作通常用于控制外接设备或指示灯，请确认硬件连接正确。

12 事件触发(PEG)

本节描述 PEG 功能相关 API 接口，支持针对指定轴根据当前位置触发单次或固定间距的脉冲事件。主要用于实现外部信号同步、标记点检测等应用场景。接口通过运动控制器轴对象进行配置和控制。

12.1 PEG-API

SetSingleEventPEG（设置单个 PEG 事件）

	Void SetSingleEventPEG(MotionController controller, AxisRef axis, int eventBegPos, int eventSelect, int? eventPulseRes = null, int? eventPulseWid = null)	
描述	设置指定轴在位置 eventBegPos 触发单次事件	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int eventBegPos	事件触发的起始绝对位置
	int eventSelect	事件选择（详情看 eventSelect）
	int? eventPulseRes	（可选）事件脉冲分辨率
		0-低（微秒）
		1-高（纳秒）
	int? eventPulseWid	（可选）事件脉冲宽度
返回值	空	
备注	配置成功后会响应相应的触发事件。	

SetEventFixedGapPEG（设置固定间隔 PEG 事件）

	void SetEventFixedGapPEG(MotionController controller, AxisRef axis, int eventBegPos, int eventGap, int eventEndPos, int eventSelect, int? eventPulseRes = null, int? eventPulseWid = null)	
描述	固定距离间隔事件触发	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int eventBegPos	事件触发的起始绝对位置
	int eventGap	事件间隔距离
	int eventEndPos	结束位置
	int eventSelect	事件选择（详情看 eventSelect）
	int? eventPulseRes	（可选）事件脉冲分辨率
		0-低（微秒）
		1-高（纳秒）

	int? eventPulseWid	（可选）事件脉冲宽度
返回值	空	
备注	设置指定轴从位置 eventBegPos 开始，至 eventEndPos，每隔 eventGap 距离触发一次事件。	

EventEnable（使能事件触发）

	void EventEnable(MotionController controller, AxisRef axis)	
描述	使能事件触发	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
返回值	空	
备注	使能指定轴的事件触发功能，将 EventOn 置 1	

注意：每一次事件触发完成后，状态都会 EventOn 都会置 0，事件触发关闭。在下一次使用要将 EventOn 打开

EventDisEnable（失能事件触发）

	void EventDisEnable(MotionController controller, AxisRef axis)	
描述	失能事件触发	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
返回值	空	
备注	失能指定轴的事件触发功能，将 EventOn 置 0	

12.2 eventSelect 说明

事件选择参数用于指定触发的事件编号或组合，具体映射如下：

eventSelect 值	含义
0	无事件
1	使用事件 #1
2	使用事件 #2
3	使用事件 #1 和 #2
4	使用事件 #3
5	使用事件 #1 和 #3
6	使用事件 #2 和 #3

eventSelect 值	含义
7	使用事件 #1、#2 和 #3
其它	未知或保留

需要现在 IO 设置成对应的事件编号。你可以在 API 调用时根据需求传递对应的 eventSelect 值。例如，若想同时触发事件 1 和事件 3，则传入 `eventSelect = 5`。

12.3 PEG-API 示例

正确示例

```
// 配置固定间隔 PEG 事件
// 开始位置 300，事件间隔 100，结束位置 1000
// 事件选择 1（使用事件#1），脉冲分辨率 1（高，纳米级），脉冲宽度 500
AAMotionAPI.SetEventFixedGapPEG(controller, AxisRef.A, 300, 100, 1000, 1,
1, 500);

// 使能 PEG 事件输出
AAMotionAPI.EventEnable(controller, AxisRef.A);

// 轴移动到 1000 位置，速度 1000
AAMotionAPI.MoveAbs(controller, AxisRef.A, 1000, 1000);

// 等待运动到位，InTargetStat 为 4 表示运动完成
while (controller.GetAxis(AxisRef.A).InTargetStat != 4);

// 配置单次 PEG 事件，位置 200，事件选择 1（使用事件#1）
// 脉冲分辨率和脉冲宽度使用默认参数
AAMotionAPI.SetSingleEventPEG(controller, AxisRef.A, 200, 1);

// 使能 PEG 事件输出，上一次 PEG 完成之后，要重新将将 PEG 使能
AAMotionAPI.EventEnable(controller, AxisRef.A);


// 回移到 0 位置，速度 1000
AAMotionAPI.MoveAbs(controller, AxisRef.A, 0, 1000);

// 等待回位完成
int iIntargetStat;
do
{
    iIntargetStat = controller.GetAxis(AxisRef.A).InTargetStat;
    Thread.Sleep(50);
} while (iIntargetStat != 4);
```


```
AAMotionAPI.SetSingleEventPEG(controller, axis, eventBegPos, eventSelect);
AAMotionAPI.EventEnable(controller, axis);    // 必须显式使能事件
AAMotionAPI.MoveAbs(controller, axis, targetPos);
```

说明

- **SetEventFixedGapPEG**：设置从 300 到 1000 区间内，每隔 100 单位距离触发事件 1，脉冲分辨率 1 和脉冲宽度 500。
- **EventEnable**：使能事件触发功能。
- **MoveAbs**：驱动轴移动至指定位置，支持位置与速度参数。
- **等待运动完成**：通过检测目标状态 `InTargetStat` 是否为 4，实现同步等待。
- **SetSingleEventPEG**：在位置 200 设置单次事件触发。
- 第二次移动回起点，并等待运动结束。

该示例展现了 PEG 功能的典型使用流程：先配置多点固定间距事件、执行运动、再配置一次性单点事件、并等待运动完成，确保事件触发的正确同步。

12.4 使用注意事项

自动复位机制
每次 PEG 事件触发完成后，系统会自动将事件使能状态 `EventOn` 置为 `0`（禁用）。

手动重使能要求
每次执行新的 PEG 运动前 **必须** 显式调用 `EventEnable()` 重新使能事件触发。

事件覆盖规则

新设置的 PEG 参数会立即生效，无需等待前序运动完成。

12.5 典型问题排查表

现象	可能原因	解决方案
事件未触发	未调用 EventEnable	确认在 MoveAbs 之前使能
偶发漏触发	运动速度过快	降低运动速度或增大脉冲宽度
脉冲变形	脉冲分辨率设置过高	调整为合适的分辨率
事件延迟	系统负载过高	适当增加线程休眠时间

13 位置锁存(Lock)

13.1 Lock-API

SetLock（配置 Lock）

```
void SetLock(MotionController controller, AxisRef axis, bool enableLock, int lockSignalPolarity, int lockSource)
```

描述	配置指定轴的锁存状态，将位置锁存（Lock）引脚设为使能或关闭状态。	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	`bool enableLock`	是否使能锁存（true 为锁存使能，false 为关闭）
	`int lockSignalPolarity`	锁存信号极性：1 表示上升沿触发，0 表示下降沿触发
	`int lockSource`	锁存源，详情看关键字 `lockSource`
返回值	空	

LockEnable（使能 Lock）

```
void LockEnable(MotionController controller, AxisRef axis)
```

描述	将目标轴的锁存功能（Lock）启用	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
返回值	空	
备注	该调用会将轴的锁存使能寄存器置为 `1`，从而激活锁存状态	

LockDisEnable（失能 Lock）

```
void LockDisEnable(MotionController controller, AxisRef axis)
```

描述	关闭目标轴的锁存功能	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
返回值	空	
备注	调用后，将锁存寄存器置为 `0`，关闭锁存状态	

13.2 Lock-API 示例

```
// 配置锁存参数：上升沿触发，源为上升沿
```

```
AAMotionAPI.SetLock(controller, AxisRef.A, true, 1, 1);

// 启用锁存(使用 SetLock 配置就可以使能 Lock 功能，那么 LockEnable 就可以不用)
//AAMotionAPI.LockEnable(controller, AxisRef.A);

// 运动到目标位置，锁存状态被触发
AAMotionAPI.MoveAbs(controller, AxisRef.A, 目标位置, 速度);

// 锁存状态保持，直到重新关闭
// 禁用锁存（示例）
AAMotionAPI.LockDisEnable(controller, AxisRef.A);
```

13.3 Lock 功能关键字说明

LockEn（锁定使能）

定义：
设置 `LockEn = 1`，启用位置锁定功能。

作用：
当位置锁定启用时，指定的输入变化（如顺时针运动上升，逆时针运动下降）会锁定当前主编码器位置，保证位置值稳定。

注意事项：
锁定和事件功能为互斥。启用锁定(`LockEn=1`)时，相关事件自动禁用，避免冲突。

LockVal（锁定值）

定义：
在输入变化期间，锁定输入变化所记录的当前位置。

详细说明：

- 对于**增量编码器**，位置由硬件直接锁定，值极为准确。
- 对于**绝对编码器**，输入变化后（在下一次采样），读取到的锁定位置会存入 `LockVal`。

更新频率：

- `LockVal` 仅在采样时间点更新一次。
- 多次有效输入变化中，仅记录最后一次变化的值。

LockCntr（锁定计数器）

定义：
记录自启用锁定（`LockEn=1`）以来，检测到的输入变化次数。

作用：
可以用作状态监控，判断输入变化的次数，甚至用于死区检测。

重置：

- 改变 LockEn 状态（如 LockEn=0 或重新赋值 0）时，LockCntr 会被重置。
- 也可通过显式赋值 0 重置计数。

LockSrc（锁定源输入）

定义：
作为锁定功能的输入信号编号。

作用：
指定哪一个输入会触发位置锁定，输入变化会反映在 LockVal 和 LockCntr 中。

拓展角色：
被指定为 LockSrc 的输入也可以在 DInMode 中配置为其他功能角色（如限位等），实现多角色共存。

示例：
输入 3 可以作为左限位开关（正常检测限位）或作为锁定触发源。

13.4 关键字使用示例

```
// 使能位置锁定
AAMotionAPI.SetLock(controller, AxisRef.A, true, 1, 3); // LockEn=1，极性上升沿，锁定源为输入 3

// 关闭位置锁定
AAMotionAPI.SetLock(controller, AxisRef.A, false, 1, 3);

// 强制重置锁定值和计数器
controller.GetAxis(AxisRef.A).LockVal = 0;    // 位置值重置
controller.GetAxis(AxisRef.A).LockCntr = 0;   // 计数器重置
```

14 力控(Force)

本章介绍力控（Force Control）相关的 API 接口及其使用方法，帮助用户实现轴的力控调节、运动及参数配置。力控功能允许通过软件设置和调节机械轴的受力状态，实现力反馈、压力控制等应用场景。用户可以在位置模式和力模式之间（开环就是电流模式）切换，调节力参数，设置目标位置和运动状态。

14.1 力控-API

GoToForceMode（进入力控模式）

	bool GoToForceMode(MotionController controller, AxisRef axis)	
描述	进入力控模式，将轴切换到力控制状态	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
返回值	无	
备注	调用后，轴进入力控状态，准备进行力调节，通常在运动之前调用	

GoToCurrMode（进入电流模式）

	bool GoToCurrMode(MotionController controller, AxisRef axis)	
描述	进入电流（或“当前”）控制模式	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
返回值	无	
备注	退出力控模式，切回电流控制	

GoToPosMode（进入位置模式）

	bool GoToPosMode(MotionController controller, AxisRef axis)	
描述	返回位置控制模式	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
返回值	无	
备注	退出力控模式，切回位置控制。	

SetPosition（设置目标位置值）

	bool SetPosition(MotionController controller, AxisRef axis, int value)

描述	设置目标位置值	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int value	目标位置值
返回值	无	
备注	设置目标位置，用于位置控制	

ForceCurr_SetForceCmdSource（设定力指令的来源）

```
void ForceCurr_SetForceCmdSource(MotionController controller, AxisRef axis, int forceCmdSrc, int[] forceCurrSlope, int[] forceCmdHTime, int[] forceCmdVal)
```

描述	设定力指令的来源参数	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int forceCmdSrc	力指令源编号（如 1 代表某信号通道）
	int[] forceCurrSlope	力指令坡度数组（调节变化快慢）
	int[] forceCmdHTime	命令持续时间数组（控制脉冲宽度）
	int[] forceCmdVal	力指令值数组（调节力大小）
返回值	无	
备注	配置用于力控的输入信号或命令源，包括力的变化速度、持续时间和大小。	

ForceCurr_SetCurrModeSWTrigSrc（设置触发源和条件）

```
void ForceCurr_SetCurrModeSWTrigSrc(MotionController controller, AxisRef axis, int currPosTh, int currPosThDir, int? forcePosErrTh = null, int? forceAInTh = null)
```

描述	设置触发源和条件，自动切换到力模式	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int currPosTh	位置阈值
	int currPosThDir	方向（1：正向，0：反向）
	int? forcePosErrTh	位置误差阈值（可选）
	int? forceAInTh	模拟输入阈值（可选）
返回值	无	
备注	设置满足条件时自动切换到力控制的触发条件。	

ForceCurr_SetForceTunePID（调节力控 PID 参数）

	void ForceCurr_SetForceTunePID(MotionController controller, AxisRef axis, int? fForceGain = null, int? fForceKi = null, int? fForceKd = null, int? fForceFFW = null, int? fForceVelFFW = null)	
描述	调节力控 PID 参数	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int? fForceGain	比例增益（Kp）
	int? fForceKi	积分增益（Ki）
	int? fForceKd	微分增益（Kd）
	int? fForceFFW	前馈力参数
	int? fForceVelFFW	速度前馈参数
返回值	无	
备注	调节力控制 PID 参数，以实现更平稳、更精准的力响应。	

ForceCurr_SetMotion（调节力控 PID 参数）

	void ForceCurr_SetMotion(MotionController controller, AxisRef axis, int absTrgt, int? relTrgt = null, int? speed = null, int? accel = null, int? decel = null)	
描述	调节力控 PID 参数	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int absTrgt	绝对目标位置
	int? relTrgt	相对目标位置（可选，null 表示不设）
	int? speed	加速度（可选）
	int? accel	减速度（可选）
返回值	无	
备注	设定目标运动的目标位置和参数，执行运动	

ForceCurr_SetSlowApproache（设置缓冲逐步逼近参数）

	void ForceCurr_SetSlowApproache(MotionController controller, AxisRef axis, int speedChgNew, int speedChgPos, int speedChgDir, int speedChgOn)	
描述	设置缓冲逐步逼近参数	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int speedChgNew	新速度

	int speedChgPos	位置阈值
	int speedChgDir	方向（1：正向，0：反向）
	int speedChgOn	开启标志（1 开启，0 关闭）
返回值	无	
备注	使轴在逼近目标位置时按照设定参数缓慢调节	

ForceCurr_SetRetractMotion（设定回退运动参数）

	void ForceCurr_SetRetractMotion(MotionController controller, AxisRef axis, int fRetractTarget, int fRetractSpeed, int? fBeginOnToPos = null)
描述	设定撤退/回退运动参数
参数	MotionController controller 控制器实例
	AxisRef axisRef 轴的标识
	int fRetractTarget 新速度
	int fRetractSpeed 位置阈值
	int? fBeginOnToPos 是否在到位后开始撤退（1：是，0：否）
	int speedChgOn 是否启用（1：启用）
返回值	无
备注	设置运动完成后的退离动作。

ForceCurr_SetPosThForAutoSwToPosMode（设置自动切换到位置模式的阈值参数）

	void ForceCurr_SetPosThForAutoSwToPosMode(MotionController controller, AxisRef axis, int posPosFlag, int posPosTh)
描述	设置自动切换到位置模式的阈值参数
参数	MotionController controller 控制器实例
	AxisRef axisRef 轴的标识
	int posPosFlag 位置阈值触发标志（通常设为 1）
	int posPosTh 位置差阈值
返回值	无
备注	当符合阈值条件时，自动切换到位置控制模式

14.2 力控 API 示例

```
// 1. 进入力控模式
AAMotionAPI.GetAxis(axis).GoToForceMode();

// 2. 设置目标位置（具体目标位置、速度可根据实际需求调整）
```

```
AAMotionAPI.GetAxis(axis).ForceCurr_SetMotion(3350, 0, 5000);

// 3. 配置缓冲逐步逼近参数
AAMotionAPI.GetAxis(axis).ForceCurr_SetSlowApproache(1000, 3000, 0, 1);

// 4. 设置 PID 参数以调节力控的响应特性
AAMotionAPI.GetAxis(axis).ForceCurr_SetForceTunePID(80000, 30, 2000, 400,
100);

// 5. 设置自动切换到力控模式的触发条件（示例阈值）
AAMotionAPI.GetAxis(axis).ForceCurr_SetCurrModeSWTrigSrc(3100, 1, 5);

// 6. 配置力指令来源及参数（示例：作用于某个力源通道）
AAMotionAPI.GetAxis(axis).ForceCurr_SetForceCmdSource(
    1,                          // forceCmdSrc: 力命令源编号
    new int[] { 2000, 2000, 2000, 2000 },    // 力指令坡度（变化速率）
    new int[] { 500, 500, 500, 500 },        // 命令持续时间（宽度）
    new int[] { 500, 200, 300, 0 }           // 命令值（大小）
);

// 7. 设置撤退（回退）动作，例如在达到目标后撤退
AAMotionAPI.GetAxis(axis).ForceCurr_SetRetractMotion(0, 20000);

// 8. 开始执行操作（提交设置）
AAMotionAPI.GetAxis(axis).Begin();

// 9. 可以在控制流程中，根据需要不断调节 PID 参数或目标位置
// 比如动态调节 PID 参数以优化响应
AAMotionAPI.GetAxis(axis).ForceCurr_SetForceTunePID(60000, 20, 1500, 300,
80);

// 10. 持续监控状态，根据实际需求调整操作
// 例如：获取当前力值和位置，判断是否达到目标等
```

说明

- **步骤 1**：切换到力控制模式，准备进行力调节。
- **步骤 2-4**：设置目标位置、缓冲参数和 PID 参数，确保力控的平稳性和响应速度。
- **步骤 5**：配置触发条件，实现自动切换到力控制状态。
- **步骤 6**：定义作用的力源（命令来源），设置力指令的坡度、时间和数值。
- **步骤 7**：设置撤退动作，确保运动完成后可以自动退出到安全状态。
- **步骤 8**：启动控制，通过调用 `Begin()` 提交所有配置。
- **步骤 9-10**：在实际运行中，可继续调整参数或读取状态，以实现精准控制。

14.3 注意事项

- **参数调整**：实际应用中，PID 参数和运动参数需要根据设备刚性、环境噪声等因素进行调试优化。
- **运动状态监控**：建议结合状态查询接口（比如位置、电流、力传感器读数）进行动态调整。
- **安全控制**：运行时应设有限制，确保在超出预设安全范围时及时退出或报警。

- **API 调用顺序：**请确保先调用 `GoToForceMode()`，再设置参数，最后调用 `Begin()`开始执行。

14.4 常见问题及解决方案

问题描述	可能原因	解决方案
力控未生效或反应迟钝	PID 参数未调优	逐步调试 PID 参数，减小响应时间或振荡
运动无反应或控制不稳定	参数配置错误	核查参数配置，确保目标位置和参数合理
自动切换条件未触发	触发条件设置不正确或阈值偏高	调整阈值，确保满足触发条件
输出力值偏离预期	力源配置不正确或输出范围有限	修改力源配置

15 矢量运动

15.1 矢量运动 API

StopVec（停止矢量运动）

	bool StopVec(MotionController controller, AxisRef axis)	
描述	停止指定轴上的矢量运动。	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
返回值	发送成功返回 true，发送失败返回 false	

VectorPause（暂停矢量运动）

	bool VectorPause(MotionController controller, AxisRef axis)	
描述	暂停指定轴上的矢量运动。	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
返回值	发送成功返回 true，发送失败返回 false	

VectorContinue（恢复矢量运动）

	bool VectorContinue(MotionController controller, AxisRef axis)	
描述	恢复暂停的矢量运动。	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
返回值	发送成功返回 true，发送失败返回 false	

VectorLinear（直线矢量运动）

	bool VectorLinear(MotionController controller, AxisRef axis, int? aPos, int? bPos, int? cPos, int? dPos, int? ePos, int? fPos, int? gPos, int? hPos, int speed, int accel, int decel, int emrgdec, int smooth)	
描述	直线矢量运动	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int? aPos	矢量运动 A 轴的绝对位置。可为空
	int? bPos	矢量运动 B 轴的绝对位置。可为空
	int? cPos	矢量运动 C 轴的绝对位置。可为空

	int? dPos	矢量运动 D 轴的绝对位置。可为空
	int? ePos	矢量运动 E 轴的绝对位置。可为空
	int? fPos	矢量运动 F 轴的绝对位置。可为空
	int? gPos	矢量运动 G 轴的绝对位置。可为空
	int? hPos	矢量运动 H 轴的绝对位置。可为空
	int speed	矢量运动速度
	int accel	矢量运动加速度
	int decel	矢量运动减速度
	int emrgdec	矢量运动急停速度
	int smooth	矢量运动平滑参数
返回值	发送成功返回 true，发送失败返回 false	
备注	由参与运动的轴中最低索引的轴发起。	

	bool VectorLinear(MotionController controller, AxisRef axis, AxisRef axisMask, int speed, int accel, int decel, int emrgdec, int smooth, params int[] pos)
描述	直线矢量运动。

参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	AxisRef axisMask	参与矢量运动位掩码
	int speed	矢量运动速度
	int accel	矢量运动加速度
	int decel	矢量运动减速度
	int emrgdec	矢量运动急停速度
	int smooth	矢量运动平滑参数
	params int[] pos	矢量运动轴绝对位置数组
返回值	发送成功返回 true，发送失败返回 false	
备注	由参与运动的轴中最低索引的轴发起。	

VectorArc（弧线矢量运动）

	bool VectorArc(MotionController controller, AxisRef axis, AxisRef axis1, AxisRef axis2, int center1, int center2, int pos1, int pos2, int dir, int circles, int speed, int accel, int decel, int emrgdec, int smooth)
描述	弧线矢量运动

参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	AxisRef axis	参与矢量弧线运动位掩码（最低轴号）

	AxisRef axis1	矢量弧线运动 X 轴
	AxisRef axis2	矢量弧线运动 Y 轴
	int center1	矢量弧线运动 X 轴中心点
	int center2	矢量弧线运动 Y 轴中心点
	int pos1	矢量弧线运动 X 轴结束位置
	int pos2	矢量弧线运动 Y 轴结束位置
	int dir	矢量弧线运动方向（0-逆时针，1-顺时针）
	int circles	矢量弧线运动附加圈数
	int speed	矢量运动速度
	int accel	矢量运动加速度
	int decel	矢量运动减速度
	int emrgdec	矢量运动急停速度
	int smooth	矢量运动平滑参数
	params int[] pos	矢量运动轴绝对位置数组
返回值	发送成功返回 true，发送失败返回 false	
备注	由参与运动的轴中最低索引的轴发起。	

SetVectorParam（配置矢量运动参数）

	void SetVectorParam(MotionController controller, AxisRef axis, int speed, int accel, int decel, int emrgdec, int smooth)

描述	用于配置轴的平滑、速度、加速度等参数	
参数	MotionController controller	控制器实例
	AxisRef axisRef	轴的标识
	int speed	矢量运动速度
	int accel	矢量运动加速度
	int decel	矢量运动减速度
	int emrgdec	矢量运动急停速度
	int smooth	矢量运动平滑参数
返回值	无	
备注	由参与运动的轴中最低索引的轴发起。	

15.2 矢量运动 API 使用实例

1. 初始化控制器

```
MotionController controller =
AAMotionAPI.Initialize(ControllerType.AGM800);
controller.ErrorOccurred += (errorCode, msgSent, errorMsg) => // 设置错误处理
{
    Console.WriteLine($"Message Sent: {msgSent}");
    Console.WriteLine($"Error Code: {errorCode}");
    Console.WriteLine(errorMsg);
};
if (!AAMotionAPI.Connect(controller))
{
    Console.WriteLine("连接控制器失败");
    return;
}
Console.WriteLine("连接成功");
```

2. 打开电机

```
AAMotionAPI.MotorOn(controller, AxisRef.A); // 打开 A 轴电机
AAMotionAPI.MotorOn(controller, AxisRef.B); // 打开 B 轴电机
```

3. 绝对位置运动

将 A、B 轴移动到位置 0，速度 10000

```
AAMotionAPI.MoveAbs(controller, AxisRef.A, 0, 10000); // A 轴移动到 0
AAMotionAPI.MoveAbs(controller, AxisRef.B, 0, 10000); // B 轴移动到 0
//等待运动完成
while (controller.GetAxis(AxisRef.A).InTargetStat != 4 ||
controller.GetAxis(AxisRef.B).InTargetStat != 4)
{
    Thread.Sleep(10);
}
Console.WriteLine("A 轴和 B 轴已到达位置 0");
```

4. 设定矢量线性运动示例

1. 单个目标位置的矢量线性运动（示例）

```
// 以 A 轴为例，目标位置为 30000，速度为 10000
AAMotionAPI.VectorLinear(
    controller,
    AxisRef.A,      // 以 A 轴为起始轴
    30000,          // 目标位置 aPos
    null,           // B 轴目标位置（留空不设置）
    null, null, null, null, null, null, // 其他轴目标都不设置
    10000,          // 运动速度
    100000,         // 加速度
    100000,         // 减速度
    1000000,        // 紧急制动减速度
    0               // 平滑系数
);
controller.GetAxis(AxisRef.A).Begin(); // 启动运动
```

2. 通过轴掩码和位置数组的矢量线性运动（示例）

假设你要同时设置 A 和 B 两个轴的目标位置，使用 AxisRef.A | AxisRef.B 作为掩码，位置数组 positions = { 10000, 5000 }，代码如下：

```
// 定义目标位置数组，按掩码中的顺序
int[] positions = new int[] { 10000, 5000 };

// 进行多轴目标位置的矢量线性运动
AAMotionAPI.VectorLinear(
    controller,
    AxisRef.A,                          // 以 A 轴为起始轴
    AxisRef.A | AxisRef.B,              // 轴掩码：同时操作 A 和 B
    10000,                              // 速度
    100000,                             // 加速度
    100000,                             // 减速度
    1000000,                            // 紧急制动减速度
    0,                                  // 平滑系数
    positions                           // 目标位置数组
);
controller.GetAxis(AxisRef.A).Begin(); // 启动运动
```

说明

- “轴掩码”AxisRef.A | AxisRef.B 表示同时控制 A 和 B 轴，它们的目标位置按照 positions 数组中的顺序对应（第一个为 A，第二个为 B）。
- 你可以根据实际的轴和运动需求，调整速度和参数。
- 确保 Begin()方法在运动命令之后调用，以启动运动。
- 目标位置数组的元素个数应与掩码中的轴个数一致，且顺序对应。

5. 暂停与继续运动

暂停 A 轴运动

```
AAMotionAPI.VectorPause(controller, AxisRef.A);
Console.WriteLine("A 轴运动暂停");
Thread.Sleep(1000); // 等待 1 秒
// 继续运动
AAMotionAPI.VectorContinue(controller, AxisRef.A);
Console.WriteLine("A 轴运动继续");
```

6. 弧线运动示例

创建一个弧线运动路径

```
AAMotionAPI.VectorArc(
    controller,
    AxisRef.A,
    AxisRef.A, // 参与弧线运动的轴（以最低索引轴为起点）
    AxisRef.B,
    5000, // center1
    5000, // center2
    0,    // pos1
    0,    // pos2
```

```
    1,     // 方向（0：顺时针，1：逆时针）
    0,     // 圆圈数
    50000, // 速度
    100000, // 加速
    100000, // 减速
    1000000, // 紧急减速
    0 // 平滑系数
);
controller.GetAxis(AxisRef.A).Begin(); // 启动弧线运动
```

7. 设置运动参数

针对某轴设置运动参数（速度、加速度等）

```
AAMotionAPI.SetVectorParam(controller, AxisRef.A, 20000, 150000, 150000,
1000000, 0);
Console.WriteLine("已设置 A 轴运动参数")
```

完整示例流程

```
// 初始化并连接
// ...（前文步骤）
AAMotionAPI.MotorOn(controller, AxisRef.A);
AAMotionAPI.MotorOn(controller, AxisRef.B);

// 移动到原点
AAMotionAPI.MoveAbs(controller, AxisRef.A, 0, 10000);
AAMotionAPI.MoveAbs(controller, AxisRef.B, 0, 10000);
while (controller.GetAxis(AxisRef.A).InTargetStat != 4 ||
controller.GetAxis(AxisRef.B).InTargetStat != 4)
{
    Thread.Sleep(10);
}
Console.WriteLine("到达原点");

// 设定线性运动
AAMotionAPI.VectorLinear(controller, AxisRef.A, 30000, 30000, null, null,
null, null, null, null, 10000, 100000, 100000, 1000000, 0);
controller.GetAxis(AxisRef.A).Begin();
Thread.Sleep(2000);

// 暂停和继续
AAMotionAPI.VectorPause(controller, AxisRef.A);
Console.WriteLine("暂停运动");
Thread.Sleep(1000);
AAMotionAPI.VectorContinue(controller, AxisRef.A);
Console.WriteLine("继续运动");

// 弧线运动
AAMotionAPI.VectorArc(controller, AxisRef.A, AxisRef.A, AxisRef.B, 5000,
5000, 0, 0, 1, 0, 50000, 100000, 100000, 1000000, 0);
controller.GetAxis(AxisRef.A).Begin();

// 等待运动结束
while (controller.GetAxis(AxisRef.A).InTargetStat !=
4||controller.GetAxis(AxisRef.B).InTargetStat != )
{
```

```
    Thread.Sleep(10);
}
Console.WriteLine("弧线运动完成");

// 关闭电机
AAMotionAPI.MotorOff(controller, AxisRef.A);
AAMotionAPI.MotorOff(controller, AxisRef.B);
```

16 关键字

16.1 轴关键字

Pos(位置)

关键词	Pos
类型	int（只读属性）
单位	用户单位（UsrUnits）设置的单位，默认为用户定义的单位，若 UsrUnits=1 则为编码器计数值。
描述	表示目标轴的当前位置，基于编码器计数，存在范围限制，范围在：-2,147,483,648 到 2,147,483,647 之间。 在重置后，Pos 值为 0。
功能	获取当前轴的位置（编码器计数或用户单位）。
限制	- 数值范围限制在 32 位有符号整数范围内。 - 超出范围会导致位置回滚（环绕循环）。 - 使用模函数(ModRev)可以避免溢出问题。
特殊说明	- 设定位置用于自主运动或校准 - UsrUnits 控制显示单位，1 表示编码器值。

使用示例

```
// 1. 初始化控制器
MotionController controller = AAMotionAPI.Initialize(ControllerType.AGM800);

// 2. 获取某轴当前位置（例如轴 A）
int currentPos = controller.GetAxis(AxisRef.A).Pos;

// 3. 输出当前位置
Console.WriteLine($"当前轴 A 位置（编码器计数）: {currentPos}");
```

AuxPos(辅助编码器位置)

关键词	AuxPos
类型	int（只读属性）
单位	用户单位（AuxUsrUnits）设置的单位；若 AuxUsrUnits=1 则以编码器计数值表示。
描述	表示辅助编码器（辅助轴或二级编码器）当前位置数值，基于编码器计数。 范围有限制，范围在：-2,147,483,648 到 2,147,483,647 之间。 在“重置”后，AuxPos 值为 0。
功能	获取辅助编码器当前位置（编码器计数或用户单位），辅助用户位置调节或校准。
限制	- 数值范围限制在 32 位有符号整数范围内。 - 超出范围会导致位置回滚（环绕循环）。 - 使用 AuxModRev 模函数防止数据溢出。

说明：

- 在电机关闭时，用户可以手动设置 AuxPos 值（用于校准）。
- 适用于辅助编码器位置检测与调节。

PDPos (脉冲方向位置)

关键词	PDPos
类型	int（只读属性）
单位	用户定义的单位（PDUsrUnits）表示的脉冲数。 如果 PDUsrUnits=1，则以脉冲为单位。
描述	报告脉冲输入位置命令读取值，反映当前位置。 重置后，值为 0。
范围	在-2,147,483,648 到 2,147,483,647 之间。当读取命令超出范围时，位置值会回滚（环绕循环）。
限制条件	如果脉冲输入超过范围，读取位置会回滚，导致位置环绕。

说明：

PDPos 报告脉冲方向输入位置命令读取，以 PD 用户单位(*PDUsrUnits*)表示。如果 *PDUsrUnits* = 1，则 PDPos 以脉冲为单位。重置时 PDPos 的值为 0。

PDPos 计数位置在 -2147483648 cnts 和 2147483647 之间。如果来自 PDPos 的命令输入超过这些限制，位置命令读取将回滚。

PDUsrUnits 仅用于报告已计数的脉冲数。也可以将输入脉冲数乘以一个因子，乘积用作参考。

参考 = $[Number\ of\ input\ pulses] * \frac{PDFact}{PDFactDen}$

显示的 PDPos 值将为

PDPos = $\mathrm{Reference} * PDUsrUnits$

示例：

PDUsrUnits = 5

(例如：5 个脉冲代表 1 毫米运动，用户希望以毫米为单位读取命令)

在输入端口输入 20 个脉冲后：

PDPos -> 4

再输入 3 个脉冲后 (共计 23 个):

PDPos -> 4

另请参阅： PDfact , PDFactDen , PDUsrUnits , PDFiltFact , PDEncFilt , PDEncDir

Vel (速度)

关键字	Vel
CAN code	5
Type	Parameter
Array Index Range	1 : 4
Access	Read only
相关轴	Yes
值类型	int（32 位）
用户单位	用户单位（UsrUnits/sec）
允许在运动状态	Yes
允许在电机开启	Yes
是否保存到存储	No
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0

Vel 数组元素及说明

索引（Index）	说明
Vel[1]	速度的滤波值（平滑处理的平均速度，可用于图表显示和噪声抑制） 滤波系数由 VelFilt 参数确定。
Vel[2]	原始速度读数（位置导数计算得到的未滤波速度值） 反映编码器实际测得的瞬时速度。
Vel[3]	16 个样本的平均速度（过去一定采样点的平均值，减少瞬时变化的影响） 基于多个样本，提供更平滑的速度估计。
Vel[4]	1/T 速度（倒数的速度值，表示速度的倒数，用于特定控制算法）

Vel [] 是一个数组，以三种不同的格式报告主编码器速度：

Vel[1] 是速度的滤波值。滤波器的因素根据 VelFilt 确定。

Vel[2] 是原始速度读数。

Vel[3] 是 16 个样本的平均速度。

以上所有均以 UsrUnits / 秒为单位。

为什么要对速度进行滤波？

原始速度读数是位置的导数。位置是一个整数值，表示读取的编码器计数的数量。可能且经常发生的是，所需的速度对应于非整数数量的编码器脉冲。

例如，如果实际速度是每个样本时间 10.5 个计数，则在 1 个样本时间后的位置将是：

$Pos_1 = 10$

（因为 0.5 个计数无法检测到）

16384 个样本每秒的计算速度为 10 * 16384 = 163840 计数/秒。

在下一个样本时间内，电机通过了 10.5 个计数。由于它从实际位置 10.5 开始，它将达到位置 21。

$Pos_2 = 21$

控制器计算位置差异以检测速度。计算出的速度是每个样本时间 11 个计数，或 11 * 16384 = 180224

我们在两个样本时间的速度读数之间收到 18384 的差异，即使实际上速度是恒定的。

对速度进行滤波有两个好处：

1. 用户在查看图表时可以看到平滑的值，并更好地指示实际的平均速度。
2. 控制也使用滤波值来消除不必要的高频噪声。

移动平均仅用于用户显示目的。它不用于控制。

AuxVel (辅助速度)

参数名	AuxVel
说明	辅助速度参数（具体用途依设备设计而定）
CAN 代码	6
类型	参数（Parameter）
访问权限	只读（Read only）
轴相关	是（Yes）
数值类型	32 位整数（int）
用户单位	辅助用户单位（Aux user units）
允许在运动中使用	是（Yes）
允许电机开启状态下使用	是（Yes）
是否存储到闪存	否（No）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0

说明：

辅助速度（AuxVel）报告辅助编码器的速度，以辅助用户单位表示。辅助速度以用户单位/秒为单位报告。

PDVel (脉冲方向速度)

参数名	PDVel
CAN 代码	7
类型	参数（Parameter）
访问权限	只读（Read only）
轴相关	是（Yes）
数值类型	32 位整数（int）
用户单位	脉冲方向用户单位（Pulse direction user units）
允许在运动中使用	是（Yes）
允许电机开启状态下使用	是（Yes）
是否存储到闪存	否（No）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0

说明：

报告脉冲方向输入的速度，单位为 PDUsrUnits/秒。是 PDPos 的导数计算得出。

IaErr (A 相电流误差)

参数名	IaErr
CAN 代码	20
类型	参数（Parameter）
访问权限	只读（Read only）
轴相关	是（Yes）
数值类型	32 位整数（int）
用户单位	毫安（mA）
允许在运动中使用	是（Yes）
允许电机开启状态下使用	是（Yes）
是否存储到闪存	否（No）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0

说明

- **IaErr** 表示相位“A”的电流误差，反映实际电流与命令电流之间的偏差，用于检测和控制电流误差。
- **示例：**如果 `IaRef = 1000`，`Ia = 990`，则 `IaErr = 100` 毫安。

IbErr (B 相电流误差)

参数名	IbErr
CAN 代码	21
类型	参数（Parameter）
访问权限	只读（Read only）
轴相关	是（Yes）
数值类型	32 位整数（int）
用户单位	毫安（mA）
允许在运动中使用	是（Yes）
允许电机开启状态下使用	是（Yes）
是否存储到闪存	否（No）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0

说明

- **IbErr** 表示相位“B”的电流误差，反映实际电流与命令电流的偏差，用于监测和控制电流差异。
- **示例：**若 `IbRef = 1000`，`Ib = 990`，则 `IbErr = 100` 毫安。

IdErr (Id 矢量电流控制误差)

参数名	IdErr
CAN 代码	22
类型	参数（Parameter）
访问权限	只读（Read only）
轴相关	是（Yes）
数值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中使用	是（Yes）
允许电机开启状态下使用	是（Yes）
是否存储到闪存	否（No）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0

说明

在矢量控制模式下（参见 ControlMode 关键字），电流控制回路在 Id 和 Iq 上闭合。 Iq 闭环以 CurrRef 作为其参考，而 Id 闭环以零作为其参考。

IdErr 是(IdRef-Id)，单位为[mA]。

这是矢量控制模式下 Id 控制回路的误差。

请注意，即使控制器不在矢量控制模式下，IdErr（与 IdRef 和 Id 一样）也会被计算。

注意：

在旧的固件版本（早于版本 1.3.0）中，Id 和 Iq 是交换的。从固件版本 1.3.0 开始，这一问题已修复，并符合上述文档。

另请参阅：IdRef, Id, IqRef, Iq, IqErr, Control Mode, CurrGain, CurrKi。

IqErr (Iq 矢量电流控制误差)

参数名	IqErr
CAN 代码	23
类型	参数（Parameter）
访问权限	只读（Read only）
轴相关	是（Yes）
数值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中使用	是（Yes）
允许电机开启状态下使用	是（Yes）
是否存储到闪存	否（No）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0

说明

在矢量控制模式下（参见 ControlMode 关键字），电流控制回路在 Id 和 Iq 上闭合。Iq 闭环以 CurrRef 为参考，Id 闭环以零为参考。

IqErr 是(IqRef-Iq)，单位为[mA]。

它是矢量控制模式下 Iq 控制回路的误差。

请注意，即使控制器不在矢量控制模式下，IqErr（就像 IqRef 和 Iq 一样）也会被计算。

注意：

在旧的固件版本（早于版本 1.3.0）中，Id 和 Iq 是交换的。从固件版本 1.3.0 开始，这一问题已修复，并符合上述文档。

另请参阅：IdRef, Id, IdErr, IqRef, Iq, CurrGain, CurrKi。

PosRef(位置参考值)

参数名	PosRef
CAN 代码	24
类型	参数（Parameter）
访问权限	只读（Read only）
轴相关	是（Yes）
数值类型	32 位整数（int）
用户单位	用户单位（UsrUnits）
允许在运动中使用	是（Yes）
允许电机开启状态下使用	是（Yes）
是否存储到闪存	否（No）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0

说明

- **PosRef** 表示内部分析器生成的位置参考，已转换为用户设定的单位。
- 该数值被直接输入到位置控制回路，用于指导运动目标。

VelRef(速度参考值)

参数名	VelRef
CAN 代码	25
类型	参数（Parameter）
访问权限	只读（Read only）
轴相关	是（Yes）
数值类型	32 位整数（int）
用户单位	用户单位（UsrUnits）
允许在运动中使用	是（Yes）
允许电机开启状态下使用	是（Yes）
是否存储到闪存	否（No）
最小值	-1,300,000,000
最大值	1,300,000,000
默认值	0

说明

- **VelRef** 表示由内部分析器生成的速度参考，已按用户单位转换，单位为 **UsrUnits/秒**。
- 该数值作为速度控制的目标输入。

CurrRef(电流参考)

方面	细节
关键字（Mnemonic）	CurrRef
CAN 编码	26
类型	参数（Parameter）
读写权限	只读（Read-only）
在运动中允许使用	是（Yes）
电机开启时允许使用	是（Yes）
是否存储到闪存	否（No）
轴相关	是（Yes）
数值范围	动态（取决于远程单元的配置）
默认值	动态（取决于远程单元的配置）
用户单位	无（No user units）
实现状态	已实现

说明：

CurrRef 表示控制回路使用的实时电流目标值，单位为毫安，具体范围由远程单元的配置决定。

IaRef(A 相电流参考)

方面	详情
关键字（Mnemonic）	IaRef
CAN 编码	27
类型	参数（Parameter）
读写权限	只读（Read only）
在运动中允许使用	否（No）
电机开启时允许使用	是（Yes）
是否存储到闪存	否（No）
轴相关	是（Yes）
数值范围	动态（取决于远程单元配置）
默认值	动态（取决于远程单元配置）
用户单位	无（No user units）
实现状态	已实现

说明：

IaRef 表示针对“相 A”施加的电流参考值，单位为毫安，范围由远程单元动态决定。

IbRef(B 相电流参考)

方面	细节
关键字（Mnemonic）	IbRef
CAN 编码	28
类型	参数（Parameter）
访问权限	只读（Read only）
在运动中允许使用	否（No）
电机开启时允许使用	是（Yes）
是否存储到闪存	否（No）
轴相关	是（Yes）
最小值	动态（依赖远程单元）
最大值	动态（依赖远程单元）
默认值	动态（依赖远程单元）
用户单位	无（No user units）
实现状态	已实现

说明

- **IbRef** 表示“相 B”的电流参考值，由控制环路实时生成，单位为毫安。
- 数值范围和默认值取决于远程控制单元的配置。

方面	细节
关键字（Mnemonic）	IdRef
CAN 编码	29
类型	参数（Parameter）
访问权限	只读（Read only）
在运动中允许使用	否（No）
电机开启时允许使用	是（Yes）
是否存储到闪存	否（No）
轴相关	是（Yes）
最小值	动态（依赖远程单元）
最大值	动态（依赖远程单元）
默认值	动态（依赖远程单元）
用户单位	无（No user units）
实现状态	固件版本 FW1.3.0 及以上支持

说明

在矢量控制模式下（参见 ControlMode 关键字），电流控制回路在 Id 和 Iq 上闭合。 Iq 闭环以 CurrRef 作为其参考，而 Id 闭环以零作为其参考。

IdRef 是 Id 控制回路的参考，单位为[mA]。实际上，它总是等于零。

请注意，即使控制器不在矢量控制模式下，IdRef 也会被计算。

注意：

在旧的固件版本（早于版本 1.3.0）中，Id 和 Iq 是交换的。从固件版本 1.3.0 开始，这一问题已修复，并符合上述文档。

另请参阅：CurrRef, Id, IdErr, IqRef, Iq, IqErr, Control Mode, CurrGain, CurrKi。

IqRef(矢量 Iq 电流参考)

方面	细节
关键字（Mnemonic）	IqRef
CAN 编码	30
类型	参数（Parameter）
访问权限	只读（Read only）
在运动中允许使用	否（No）
电机开启时允许使用	是（Yes）
是否存储到闪存	否（No）
轴相关	是（Yes）
最小值	动态（依赖远程单元）
最大值	动态（依赖远程单元）
默认值	动态（依赖远程单元）
用户单位	无（No user units）
实现版本	1.3.0

说明

在矢量控制模式下（参见 ControlMode 关键字），电流控制回路在 Id 和 Iq 上闭合。Iq 闭环以 CurrRef 为参考，Id 闭环以零为参考。

IqRef 是 Iq 控制回路的参考，单位为[mA]。它实际上总是等于 CurrRef。

请注意，即使控制器不在矢量控制模式下，IqRef 也会被计算。

注意：

在旧的固件版本（早于版本 1.3.0）中，Id 和 Iq 是交换的。从固件版本 1.3.0 开始，这一问题已修复，并符合上述文档。

另请参阅：CurrRef, IdRef, Id, IdErr, Iq, IqErr, 控制模式, CurrGain, CurrKi。

ConFlt（控制器故障码）

方面	细节
关键字（Mnemonic）	ConFlt
CAN 编码	31
类型	参数（Parameter）
访问权限	可在运动中读取（Yes）
电机开启时允许使用	可（Yes）
是否存储到闪存	否（No）
轴相关	是（Yes）
数值范围	-2,147,483,648 至 2,147,483,647（int 范围）
默认值	0
用户单位	无（No user units）
实现状态	已实现

ConFlt 报告导致电机禁用的错误。每个输入到 ConFlt 的故障也会记录在 ErrLog 中。
ConFlt 在下次电机使能时被清除。
下表显示了 ConFlt 的值及其含义：

值	含义
0	没有发生故障
1001	检测到中止信号
1002	电机相位对地短路
1003	编码器断开连接
1004	FPGA 看门狗未接收到
1005	PWM 死区时间过短
1006	霍尔输入断开连接
1007	电机堵转
1008	总线电压过高
1009	总线电压过低
1010	逻辑电压过高
1011	逻辑电压过低
1012	总线电流过高
1013	相 A 电流过高
1014	相 B 电流过高
1015	相 C 电流过高
1016	电机电流过高
1017	驱动器功率超出限制
1018	IPM 温度过高
1019	速度过高
1020	位置误差超出限制

1021	5V 输出引脚之一用于编码器或 I/O 激活了电流限制。检查您的硬件连接！
1022	电机过热
1023	电源电压过低
1024	电机电流过高
1025	驱动器功率超出限制
1026	IPM 温度过高
1027	速度过高
1028	位置误差超出限制
1029	编码器故障
1030	电机过载
1031	电机温度过高
1032	电机电流不平衡
1033	电机电压不平衡
1034	电机相位不平衡
1035	电机相位丢失
1036	电机相位短路
1037	电机相位对地短路
1038	电机相位对相位短路
1039	电机相位对电源短路
1040	电机相位对逻辑短路
1041	电机相位对地短路
1042	电机相位对相位短路
1043	电机相位对电源短路
1044	电机相位对逻辑短路
1045	电机相位对地短路
1046	电机相位对相位短路
1047	电机相位对电源短路
1048	其他成员轴关闭了电机
1049	全闭环路速度差异过大
1050	外部故障输入（例如：外部驱动器）被激活
1051	内部继电器仍然关闭。尝试在电源开启后等待更长时间再启用电机
1052	STO2 被激活
1053	AmpType 值不允许用于此产品
1054	至少一个所需的交流电源输入相位被切断

说明

- ConFlt 存储当前故障码，故障由不同的数值代表不同的原因。
- 这个值在故障发生时被设置，清除在下一次电机启用后。
- 具体故障码的含义详见表中的详细列表，包括硬件故障、通讯中断、电压过高等。

相关信息

- ErrLog：每个故障码都可以在 ErrLog 中找到详细记录。

MotionStat（运动状态）

项目	内容
关键字（Mnemonic）	MotionStat
CAN 编码	32
类型	参数（Parameter）
访问	只读（Read only）
在运动中允许	允许（Yes）
电机开启时允许	允许（Yes）
是否存储到闪存	否（No）
轴相关	是（Yes）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0
用户单位	无（No user units）
实现状态	已实现

MotionStat 报告当前运动的状态。MotionStat 中的每个位代表一个运动状态。在某些情况下，可能会有多个位被激活。当电机不运动时，MotionStat = 0。下表显示了 MotionStat 位的含义：

0	正在运动
1	等待中（重复运动期间）
2	在重复停止中（跟随 StopRep 命令）
3	请求停止
4	加速中
5	减速中
6	等待平滑结束
7	在 ECAM 停止中（跟随 StopECAM 命令）
8	在 FIFO 停止中（跟随 StopFIFO 命令）
9	等待输入（运动暂停，直到用户定义输入的上升沿）
10	CNCA 成员
11	现在涉及 CNCA
12	在 CNCA 停止中

StatReg（状态寄存器）

名称	值类型
CAN code	33
Type	参数（Parameter）
Access	只读
轴相关	是
Value type	int（32-bit）
User units	无
允许在运动中	是
允许在电机开启时	是
保存到闪存	否
Min value	-2,147,483,648
Max value	2,147,483,647
Default value	0

StatReg 是一个状态寄存器。StatReg 的位显示以下状态：

位	状态
0	换向完成位
1	再生已开启
2	保留
3	超过最大总线电压 (MaxVBus)
4	低于最小总线电压 (MinVBus)
5	主继电器已开启（适用于带主继电器的产品）
6	绝对最大总线电压超出 (MaxVBusAbs)
8..7	总线电压警告： 00 – 无警告。 总线电压深处于其范围内。 01 – 低警告。 VBus ε (0.88 * MaxVBus)，或 VBus δ (1.12 * MinVBus) 10 – 中等警告。 VBus ε (0.92 * MaxVBus)，或 VBus δ (1.08 * MinVBus) 11 – 高警告。 VBus ε (0.96 * MaxVBus)，或 VBus δ (1.04 * MinVBus) 位 7 是最低有效位。
10..9	电流参考警告： 00 – 无警告。 电流参考 (CurrRef) 深处于其范围内。

	01 – 低警告。 Abs(CurrRef) > (0.88 * PeakCL) 10 – 中等警告。 Abs(CurrRef) > (0.92 * PeakCL) 11 – 高警告。 Abs(CurrRef) > (0.96 * PeakCL) 位 9 是最低有效位。
12..11	电源单元温度警告： 00 – 无警告。 PwrTemp 在其范围内。 01 – 低警告。 PwrTemp > (0.88 * MaxPwrTemp) 10 – 中等警告。 PwrTemp > (0.92 * MaxPwrTemp) 11 – 高警告。 PwrTemp > (0.96 * MaxPwrTemp) 位 11 是最低有效位。
14..13	通用控制饱和警告： 这些警告位指示控制算法在以下饱和状态中的深度： 速度饱和 (MaxVel) 或电流饱和 (PeakCL 或 ContCL 当连续电流限制激活时，或通过模拟输入激活时的限制，或通过固定值激活时的限制) 或电压饱和 (MaxPWM)。 如果在给定的控制周期（采样）中激活了任何这些饱和状态（最大或最小值），计数器将增加。如果在 1 秒内此计数器高于以下定义的阈值，则在此状态位上发出适当的警告状态。 此警告状态每秒更新一次，并保持不变 1 秒（因此用户可以轻松观察到 PC Suite 上的相关 LED）。 此警告极为重要，因为一旦控制算法进入其饱和状态，意味着跟踪性能受到某种程度的限制（取决于饱和的深度）。 饱和深度的测量是指在 1 秒内至少有一个饱和状态被激活的时间段。例如，100 毫秒的饱和深度意味着在 1 秒内至少有 100 毫秒的累计时间内，控制算法达到了一个饱和状态。这意味着：10%的控制周期处于饱和状态（其中之一，不一定是所有周期中的同一个），或者至少 1638 个周期（通常 Agito 控制器的采样频率为 16384 Hz）。 请注意，这里的警告从非常低的饱和深度开始发出。这是因为饱和是运动性能的关键状态，有时用户可能没有意识到饱和状态。通过这些状态位，可以轻松注意到饱和事件。 00 – 无警告。 饱和深度小于 0.2%。 01 – 低警告。 饱和深度大于 0.2%。 10 – 中等警告。 饱和深度大于 1%。 11 – 高警告。 饱和深度大于 5%。 位 13 是最低有效位。 请注意，位 21 到 23，如下所述，提供了每个饱和状态（速度、电流和电压）的瞬时状态。
16..15	电机温度警告： 00 – 无警告。 MotorTemp 在其范围内。

	01 – 低警告。 MotorTemp > (0.88 * MaxMotorTemp) 10 – 中等警告。 MotorTemp > (0.92 * MaxMotorTemp) 11 – 高警告。 MotorTemp > (0.96 * MaxMotorTemp) 位 15 是最低有效位。 （这些位仅在 FW 版本 1.4.0 及以上使用）
17	反向硬件限位已激活。这与 LimitsStat 的状态相同。
18	正向硬件限位已激活。这与 LimitsStat 的状态相同。
19	位置参考（PosRef）小于软件位置前限（RevPLim）。
20	位置参考（PosRef）大于软件位置前限（FwdPLim）。
21	电流饱和已激活。如果 abs(CurrRef)大于以下之一：PeakCL 或 ContCL（当连续电流限制激活时），或通过模拟输入激活的限制，或通过固定值激活的限制，则电流饱和已激活。 请注意，与上面描述的位 14..13 不同，此位反映了饱和的瞬时状态（在最近的控制周期中是否激活）。
22	电压饱和已激活。电压饱和指的是 Va 或 Vb 或 Vc，与±MaxPWM 相比。 请注意，与上面描述的位 14..13 不同，此位反映了饱和的瞬时状态（在最近的控制周期中是否激活）。
23	速度饱和已激活。速度饱和指的是 VelRef，与±MaxVel 相比。 请注意，与上面描述的位 14..13 不同，此位反映了饱和的瞬时状态（在最近的控制周期中是否激活）。
25..24	其他警告。 这是一个通用警告，用于以下各种目的： 00 – 无警告。 01 – 低警告。 目前未使用。 10 – 中等警告。 I^2t 电流限制已激活。 11 – 高警告。 目前未使用。 位 24 是最低有效位。
26	用户修改了速度控制回路或位置控制回路中至少一个高阶（双二次）滤波器的定义。 当以下参数之一被赋予新值时，此位被设置：VelFiltDef[]，VelFiltOn[]，PosFiltDef[]或 PosFiltOn[]。 此位反映了一个无效状态，其中滤波器的定义与滤波器的内部系数不相关。 请注意，结果是，在 CalcFilters 成功执行之前，控制器将不允许启用轴（MotorOn=1）。
27	CalcFilters 执行失败。这意味着双二次滤波器的内部系数与滤波器的定义不匹配。 请注意，结果是，在 CalcFilters 成功执行之前，控制器将不允许启用轴（MotorOn=1）。
28	动态制动状态位。当动态制动激活时，它被设置为“1”。
29	静态制动锁定请求。当静态制动被命令为锁定状态时，它被设置为“1”，当静态制动被（请求）释放时，它被设置为“0”。
30	回原点开关已启用。仅当离散输入被编程为此轴的回原点输入时有效。

“警告”位，如上所述，可用于用户识别“几乎”故障的情况。

请注意，“Agito PC Suite”软件使用此状态参数的位值来控制其状态面板上的 LED。 警告状态有四种值（无、低、中和高），在 PC Suite 状态面板上反映为多色 LED（关闭、黄色、橙色和红色，分别对应）。

另请参阅：

MotionStat, MotionReason, ConFlt, MotorOn , VBus, MaxVBus, MinVBus, CurrRef, PeakCL, PwrTemp, MaxPwrTemp, MaxVel, ContCL, CurrLimMode, CurrLimFwd, CurrLimRev, LimitsStat, RevPLim, FwdPLim, PosRef, Va, Vb, Vc, MaxPWM, VelRef, PeakTime, VelFiltDef[], VelFiltOn [], PosFiltDef[], PosFiltOn[] 和 .CalcFilters

MasterPos（齿轮主位置）

项 目	内容
CAN 编码	44
类型	参数（Parameter）
访问权限	只读（Read only）
允许在运动中	是（Yes）
允许在电机开启时	允许（Yes）
是否存储到闪存	否（No）
轴相关	是（Yes）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0
用户单位	以用户单位显示（In user units）
实现状态	已实现（Implemented）
关联参数	详见：MasterFact, MotionMode

说明

- **MasterPos** 返回的值是在齿轮传动配置中的主轴位置，用于闭环控制和位置同步。
- 它是一个只读参数，用户无法修改，但可以在运动中读取。
- 适合用来监控或同步多轴运动中的主轴位置。

项 目	内容
关键字（Mnemonic）	IndexStat
CAN 编码	45
类型	参数（Parameter）
访问权限	只读（Read only）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
是否存储到闪存	否（No）
轴相关	是（Yes）
最小值	0
最大值	1
默认值	0
用户单位	无（No user units）
实现状态	已实现（Implemented）

说明

IndexStat 保存了主编码器索引输入的当前状态（“0”或“1”）（仅适用于增量和 SinCos 编码器）。

如果需要在应用中检测编码器的索引事件，可以定期读取该参数值以实现响应操作。

另请参阅：

StopOnIndex , IndexPos , AuxIndexStat 和 AuxIndexPos 。

项 目	内容
关键字（Mnemonic）	AuxIndexStat
CAN 编码	46
类型	参数（Parameter）
访问权限	只读（Read only）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
是否存储到闪存	否（No）
轴相关	是（Yes）
最小值	0
最大值	1
默认值	0
用户单位	无（No user units）
实现状态	已实现（Implemented）

说明

AuxIndexStat 用以反映辅助编码器的索引信号当前状态。

值为“1”：索引信号被触发。

值为“0”：未触发。

主要用于辅助编码器的校准或特殊同步需求。

由于是只读参数，用户可以用它监测索引信号的实时状态，但不能用来修改。

如需要实现位置校准或监控辅助索引状态时，可以定期读取此参数值。

另请参阅：

AuxIndexPos, IndexStat 和 IndexPos。

项 目	内容
关键字（Mnemonic）	IndexPos
CAN 编码	47
类型	参数（Parameter）
访问权限	只读（Read only）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
是否存储到闪存	否（No）
轴相关	是（Yes）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0
用户单位	以用户单位表示（In user units）
实现状态	已实现（Implemented）

说明

IndexPos 保存了检测到编码器索引的最后一个主编码器位置。

IndexPos 仅适用于增量和 SinCos 编码器。

请注意，为了正确检测索引信号及其位置，编码器速度必须小于控制器采样频率，以便索引脉冲宽度至少与单个采样一样宽。

另请参阅：

StopOnIndex , IndexStat , AuxIndexStat 和 AuxIndexPos 。

项 目	内容
关键字（Mnemonic）	AuxIndexPos
CAN 编码	48
类型	参数（Parameter）
访问权限	只读（Read only）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
是否存储到闪存	否（No）
轴相关	是（Yes）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0
用户单位	附加用户定义单位（Aux user units）

说明

AuxIndexPos 保存了检测到辅助编码器索引的最后一个辅助编码器位置。

AuxIndexPos 仅适用于增量编码器。

请注意，为了正确检测索引信号及其位置，编码器速度必须小于控制器采样频率，以便索引脉冲宽度至少与单个采样一样宽。

另请参阅：

AuxIndexStat, IndexStat 和 IndexPos。

LimitsStat（限位状态）

项 目	内容
关键字（Mnemonic）	LimitsStat
CAN 编码	49
类型	参数（Parameter）
访问权限	只读（Read only）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
是否存储到闪存	否（No）
轴相关	是（Yes）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0
用户单位	无（No user units）
实现状态	已实现（Implemented）

详细说明

LimitsStat 参数是一个整型数值，存放了两个重要的限位输入的状态。

Bit 0（最低有效位）：代表逆限位开关（RLS）的状态：

- 0：未激活
- 1：激活（触发限位）

Bit 1：代表正限位开关（FLS）的状态：

- 0：未激活
- 1：激活

其他位未定义或保留。可以通过按位与操作判断具体的限位状态。

例：如果 LimitsStat 为 3（二进制 11），表示两个限位都激活。

该参数为只读，用户可以用来监控当前机械是否达到限位状态，确保运动安全。

如果需要在运动控制中监控限位状态，读取此参数即可获知实时情况。

项 目	内容
关键字（Mnemonic）	MotorType
CAN 编码	50
类型	参数（Parameter）
访问权限	读写（Read / Write）
允许在运动中	否（No）
允许在电机开启时	否（No）
是否存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	0
最大值	7
默认值	0（未知）
用户单位	无（No user units）
实现状态	已实现（Implemented）

MotorType 是连接到控制器的电机类型。此变量决定如何将电压施加到电机上。选择错误的 MotorType 可能会导致严重后果。

可以分配给 MotorType 的值有：

值	电机类型
0	未知
1	直流有刷
2	音圈
3	线性直流无刷
4	旋转直流无刷
5	模拟

每种电机类型的含义如下所述：

未知

这是新控制器的默认值。在用户设置其他有效电机类型之前，不会将电压施加到功率级输出。

直流有刷

如果设置了直流有刷电机类型，则将全电压施加到两个电源端子。请参阅硬件手册以了解使用哪个端子。

音圈

这种电机类型与直流有刷相同。

线性/旋转直流无刷

电压施加到三个电机相位，并使用换向来确定施加到每个电机电源端子的电压。类型之间的区别实际上在于反馈的配置和电机极的处理。分开的类型有助于在 PCSuite 中进行配置。

请参阅硬件手册以获得正确的相位连接。

模拟

这种电机类型用于开发期间的模拟。它允许生成模拟的轮廓、输入和输出行为，而无需实际将电机连接到控制器。

ContCL（连续电流限制）

项 目	内容
关键字（Mnemonic）	ContCL
CAN 编码	51
类型	参数（Parameter）
访问权限	读写（Read / Write）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
是否存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	动态（依据远程单元）——实际值由控制系统动态决定
最大值	动态（依据远程单元）
默认值	动态（依据远程单元）
用户单位	无（No user units）

注意事项及说明

ContCL（连续电流限制，以[mA]为单位）与 PeakCL 和 PeakTime 参数一起使用，以定义放大器/电机的 I^2t 功率限制方案的行为。

- **动态值**：由于值依赖于远程硬件/单元，实际设置会动态变化，需根据系统具体配置调整。
- **作用**：与 PeakCL 和 PeakTime 一起，限制电机在持续工作时的电流，以避免过热或损坏。

相关参数及参考

- **PeakCL**：峰值电流限制。
- **PeakTime**：峰值持续时间。
- **MotorCurr**：当前电流值。
- **StatReg**：状态寄存器参数。

PeakCL（峰值电流限制）

项 目	内容
关键字（Mnemonic）	PeakCL
CAN 编码	52
类型	参数（Parameter）
访问权限	读/写（Read / Write）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
是否存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	动态（依赖于远程单元，实际值由硬件控制）
最大值	动态（依赖于远程单元，实际值由硬件控制）
默认值	动态（依赖于远程单元）

基本上， PeakCL 是峰值电流限制 （单位：[mA]） 。电流命令 （CurrRef） 永远不会超过此值（ ± PeakCL 用于饱和 CurrRef） 。

PeakCL 的最大允许值（见上表）是该产品的最大允许电流。

请注意，PeakCL 在正弦换向时，指的是电流正弦信号的峰值，而不是其均方根值。

然而，PeakCL 参数与 ContCL 和 PeakTime 参数一起被控制器用于实现 I^2t 放大器/电机功率限制方案。

PeakCL、ContCL 和 PeakTime 的最大值定义了 I^2t 限制方案，以确保不超过放大器的能力。然而，用户可以修改这些参数以定义一个新的限制方案，例如匹配电机的功率限制。

例如，某个产品可以有如下最大值：

PeakCL = 16000

ContCL = 8000

PeakTime = 1000

这意味着该产品可以在 1000 毫秒（1 秒）内驱动高达 16000mA（16A）的电流，其连续电流限制为 8000mA（8A）。这是产品本身的限制。

然而，对于某个电机/应用，用户可以设置：

PeakCL = 5000

ContCL = 3000

PeakTime = 500

以不超过电机的功率能力，或者如果应用没有提供额定冷却条件，也不超过放大器的功率能力。

放大器/电机 I2t 功率限制方案如何工作？

此限制使用以下参数：ContCL（连续电流限制）、PeakCL （峰值电流限制） 和 PeakTime （最大允许峰值电流时间） 。

限制代码始终计算 I^2（使用整体电机电流：MotorCurr）随时间变化 （"积分" I^2） 。如果它变得高于 $ContCL^2$，则激活限制。

一旦激活限制，CurrRef 通常被限制（饱和）在 ± PeakCL，现在被限制在 ±ContCL，因此实际上放大器现在被限制为提供不超过 ContCL 的电流。

在任何时候，I^2随时间计算，使用一阶滤波器模拟电机的热量散发 （或者更确切地说，电机的温度） 。 这样，I^2随时间"积分"以估计放大器/电机内产生的温度。一阶滤波器的系数是 PeakTime、PeakCL 和 ContCL 的函数，因此：

在长时间没有电流到电机后，如果现在驱动一个 PeakCL 电流（$PeakCL^2$是滤波器的输入），在 PeakTime 时间内，滤波器的输出将达到 $ContCL^2$值，这将触发限制并将电流限制为 ContCL。

当然，如果驱动到电机的电流低于 PeakCL，则在不触发限制的情况下可用的时间会更长。 实际上，如果 电流等于或低于 ContCL，则限制将永远不会被激活。

请注意，PeakCL 电流的可用性，在 PeakTime 时间内，仅在长时间没有电流到电机后（因此滤波器输出为 0）。 在其他情况下（例如：长时间的 0.9*ContCL），实际的峰值时间将（大大）缩短，因为滤波器输出将更快地达到 $contCL^2$（这相当于电机的温度行为，因为在这种情况下，由于非零电流随时间的存在，电机已经被加热）。

一旦 I^2随时间低于 0.9 x $ContCL^2$，限制被解除。这是为了创建一个滞后，以便限制不会在连续电流周围快速开/关/开/关。

请注意，此限制的所有参数都可以动态修改。

请注意，限制仅适用于 CurrRef 饱和，通常是 PeakCL。它对控制器支持的其他限制没有影响：使用模拟输入的电流限制或固定值（参见：CurrLimMode）。

请注意，虽然用户可以设置值如 PeakCL=100 和 ContCL = 200（即峰值 < 连续或峰值=连续），但这种情况是无效的，控制器将内部使用不同的连续电流值，等于 PeakCL/2。 一旦用户正确设置了两个参数，控制器将返回使用 PeakCL 和 ContCL 的值。

限制状态反映在 StatReg 参数中，第 24,25 位。在 PCSuite 中，当限制激活时，您将看到"其他警告"LED 变为"橙色"。

注意：
此 I^2t 功率限制方案仅在电流控制回路激活时工作（参见 ControlMode 参数） 或者如果使用外部放大器并且 CurrRef 用于驱动模拟输出。
在后一种情况下，CurrRef 用于此限制的方程中，而不是 MotorCurr（见上面的解释）。
另请参阅：

ContCL, PeakTime , MotorCurr 和 StatReg。

注意事项

- **参数动态调整**：这些参数可以在运行时调整，以匹配不同负载和散热条件。
- **参数关系**：如果设置 PeakCL 低于 ContCL，系统会自动调整，实际连续电流会变成 PeakCL 的 1/2，以确保安全。
- **限制影响**：该限制只影响峰值指令（CurrRef）饱和，不作用于其他硬件或模拟限制（如模拟输入限制）。
- **使用前提**：确保电流控制已激活（ControlMode 参数）或外部放大器已启用。

项 目	内容
关键字（Mnemonic）	PeakTime
CAN 编码	53
类型	参数（Parameter）
访问权限	读/写（Read / Write）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
是否存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	1 毫秒
最大值	3000 毫秒（即 3 秒）
默认值	500 毫秒
用户单位	无（No user units）

详细说明

作用：PeakTime 定义在峰值电流（PeakCL）开始后，允许持续多长时间（最大峰值电流 dürfen 在此时间内施加），超过后将触发热功率限制，实际限制在 ContCL（连续电流）水平。

功能关系：

- 结合 PeakCL 和 ContCL，控制器利用 I^2t 算法对电机功率进行热管理。
- PeakTime 越长，允许在峰值电流下运行的时间越久，减缓对热的限制。

影响：

- 在峰值电流峰值持续时间内（不超过 PeakTime），电流命令可达 PeakCL。
- 如果超过 PeakTime，限制将激活，电流将被限制在 ContCL 范围内。

实际应用：

- 用户可以设定 PeakTime 以平衡性能与热保护。

项 目	内容
关键字（Mnemonic）	PolePrs
CAN 编码	54
类型	参数（Parameter）
允许在运动中	否（No）
允许在电机开启时	否（No）
是否存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	1
最大值	50
默认值	4
用户单位	无（No user units）

注意事项与说明

PolePrs 是受控电机每机械转一圈的磁极对数。此数字与 EncRes 一起用于确定施加到电机相位的电压模式。请使用电机数据表输入正确的 PolePrs 值。

对于线性电机，输入 PolePrs = 1, EncRes = 每个电周期的编码器计数数

警告：

如果 PolePrs 错误，将会发生意外行为。这可能导致控制器、电机或连接到电机的任何其他系统部件严重损坏。

另请参阅：

- **EncRes**：编码器分辨率（每电周期的脉冲数），与 PolePrs 结合使用。

EncType（编码器反馈类型）

项 目	内容
关键字（Mnemonic）	EncType
CAN 编码	55
类型	参数（Parameter）
访问权限	只读（No）
允许在运动中	否（No）
允许在电机开启时	否（No）
是否存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	0
最大值	6
默认值	1（增量式编码器）
用户单位	无（No user units）

EncType 设置连接到受控电机的编码器反馈类型。可以分配给 EncType 的值如下所示。请注意，这些编码器类型中的某些在所有产品上不可用，有些需要特殊选项。

值	编码器类型
0	未知
1	增量编码器
2	Sin/Cos
3	Endat
4	SSI
5	Nikon 17 位编码器

相关参数

- **EncRes**：编码器分辨率（脉冲数或分辨率，需配合设置以获得精准反馈）。

EncRes（编码器计数）

项 目	内容
关键字（Mnemonic）	EncRes
CAN 编码	56
类型	参数（Parameter）
访问权限	只读（No）
允许在运动中	否（No）
允许在电机开启时	否（No）
是否存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	1
最大值	2,000,000
默认值	1
用户单位	无（No user units）

EncRes 是机械旋转中编码器计数的数量。此数字与 PolePrs 一起用于确定施加到电机相位的电压模式。请使用电机数据表输入正确的 EncRes 值。

对于线性电机，输入 PolePrs = 1，EncRes = 每个电周期的编码器计数。

警告：

如果 EncRes 错误，将会发生意外行为。这可能导致控制器、电机或连接到电机的任何其他系统部件严重损坏。

相关参数

- **PolePrs**：每机械旋转中的磁极对数
- **EncType**：编码器类型（比如增量式、Sin/Cos 等）

项 目	内容
关键字（Mnemonic）	EncFilt
CAN 编码	57
类型	参数（Parameter）
访问权限	读/写（Yes）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
是否存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	0
最大值	255
默认值	10

指定要应用于增量主编码器输入通道（A、B 和 Index）的数字滤波器。

滤波器由硬件实现。

适用于 AG300 控制器

滤波器的特征如下：

- 输入由滤波器机制采样。只有在连续 6 次采样具有相同值后，输入电平才被确认。这意味着信号的“1”逻辑需要至少采样 6 次，同样“0”逻辑也需要采样，总共（2 * 6）次采样。
- 滤波器频率（采样频率）由 EncFilt 参数决定。
- 如果 EncFilt == 0，滤波器频率等于 300MHz，即 DSP 的时钟频率。
- 如果 EncFilt 不等于 0，滤波器频率为：

$$Filter\ Frequency = \frac{300}{2 \cdot EncFilt} \quad [MHz]$$

- 因此，A、B 和 Index 信号的最大理论输入频率（在没有噪声的情况下）可以使用以下公式计算：

$$Max\ Input\ Frequency = \frac{FilterFrequency}{2 * 6} \quad [MHz]$$

$\therefore if\ EncFilt = 0,$

$$Max\ Input\ Frequency = \frac{300}{2 \cdot 6} \quad [MHz]$$

$Else,$

$$Max\ Input\ Frequency = \frac{300}{(EncFilt \cdot 2)(2 \cdot 6)} \quad [MHz]$$

- 也许更适用的是轴的相关最大速度（在 x4 计数/秒之后），如下所示：

$$Max\ Axis\ Speed = \frac{FilterFrequency}{3} \quad [\frac{counts}{sec}]$$

∴ *如果* $EncFilt = 0$ ：

$$Max\ Axis\ Speed = \frac{300 * 10^6}{3} \quad [\frac{counts}{sec}]$$

Else：

$$Max\ Axis\ Speed = \frac{300 * 10^6}{EncFilt \cdot 6} \quad [\frac{counts}{sec}]$$

注意：

- EncFilt 设置的“最大输入频率”（或最大轴速度）是输入频率（轴速度）的理论上限，假设理想的方波信号（无限斜率）和理想的电子设备（控制器中接收芯片没有延迟和斜率）。实际的“最大输入频率”会更小，取决于信号的质量。
- 此外，如果信号上的噪声会“干扰”滤波器计数 6 次连续采样的给定逻辑电平。因此，建议不要将 EncFilt 设置得太高。
- 总之，我们建议将 EncFilt 设置为计算出的理论最大轴速度是系统中实际最大速度的两倍。
- 在给定系统中设置 EncFilt 的实际值可以根据上述公式/考虑进行初步设置，但必须在系统中仔细测试和调整，以适应编码器信号上出现的噪声和信号本身的质量。
- 从滤波器的角度来看，最快的输入信号可以是 25MHz（EncFilt = 0 且假设没有噪声），这与最大速度 100 x 10^6 计数/秒相关。
- 最快的滤波器频率是 300 MHz（EncFilt = 0），最慢的滤波器频率约为 0.59 MHz（EncFilt 的最大值为 255）。

滤波器的特征如下：

- 输入由滤波器机制采样。只有在连续 4 次采样具有相同值后，输入电平才被确认。这意味着信号的“1”逻辑需要至少采样 4 次，同样“0”逻辑也需要采样，总共（2 * 4）次采样。

- 滤波器频率（采样频率）由 EncFilt 参数决定。

$$Filter\ Frequency = \frac{100}{2^{EncFilt+1}} \quad [MHz]$$

- 因此，编码器信号 A、B 和 Index 的最大理论输入频率（在没有噪声的情况下）可以使用以下公式计算：

$$Max\ Input\ Frequency = \frac{100}{2^{EncFilt+1} \cdot (2 \cdot 4)} \quad [MHz]$$

或通过此表：

EncFilt	滤波器时钟	最大编码器频率
0	CLK/2 (50MHz)	CLK/16 (6.25MHz)
1	CLK/4 (25MHz)	CLK/32 (3.125MHz)
2	CLK/8 (12.5MHz)	CLK/64 (1.5625MHz)
3	$12.5 * 10^6$	
4	$6.25 * 10^6$	
5	$781.25 * 10^3$	
6	$390.625 * 10^3$	
7	$195.3125 * 10^3$	

注意：

- EncFilt 设置的“最大输入频率”（或最大轴速度）是输入频率（轴速度）的理论上限，假设理想的方波信号（无限斜率）和理想的电子设备（控制器中接收芯片没有延迟和斜率）。实际的“最大输入频率”会更小，取决于信号的质量。

- 此外，如果信号上的噪声会“干扰”滤波器计数 6 次连续采样的给定逻辑电平。因此，建议不要将 EncFilt 设置得太高。

- 总之，我们建议将 EncFilt 设置为计算出的理论最大轴速度是系统中实际最大速度的两倍。

- 在给定系统中设置 EncFilt 的实际值可以根据上述公式/考虑进行初步设置，但必须在系统中仔细测试和调整，以适应编码器信号上出现的噪声和信号本身的质量。

- 从滤波器的角度来看，最快的输入信号可以是 6.25MHz，意味着 $25 \cdot 10^6$ 计数/秒（EncFilt = 0 且假设没有噪声）。如果需要更快的输入，请咨询 Agito。

- 最快的滤波器频率是 50 MHz（EncFilt = 0），最慢的滤波器频率是 781.25 KHz（EncFilt 的最大值为 7）。

EncDir（编码器方向）

项 目	内容
关键字（Mnemonic）	EncDir
CAN 编码	58
类型	参数（Parameter）
访问权限	只读（No）
允许在运动中	否（No）
允许在电机开启时	否（No）
是否存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	0
最大值	1
默认值	0（代表默认正向，即正向为顺时针旋转时，编码器值增大）

EncDir 可用于更改编码器读取的方向。如果编码器读取在电机顺时针旋转时向正方向前进，并且 EndDir = 0，则设置 EncDir = 1 以使读取在电机顺时针旋转时向负方向前进。

AuxEncType（辅助编码器类型）

项 目	内容
关键字（Mnemonic）	AuxEncType
CAN 编码	59
类型	参数（Parameter）
访问权限	只读（No）
允许在运动中	否（No）
允许在电机开启时	否（No）
是否存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	0
最大值	6
默认值	1（增量式编码器）

AuxEncType 设置辅助编码器反馈类型。可以分配给 AuxEncType 的值列在下面。请注意，这些编码器类型中的一些在所有产品上都不可用，有些需要特殊选项。

值	编码器类型
0	未知
1	增量编码器
2	正弦/余弦
3	Endat
4	SSI
5	Nikon 17 位编码器
6	绝对 BiSS-C
7	模拟位置反馈
8	多摩川

相关参数

AuxEncFilt（辅助编码器数字滤波器）

项 目	内容
关键字（Mnemonic）	AuxEncFilt
CAN 编码	60
类型	参数（Parameter）
访问权限	读/写（Yes）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
是否存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	0
最大值	255
默认值	0

AuxEncFilt 定义了辅助增量编码器输入信号（A、B 和 Z）的硬件数字滤波器的行为。

有关滤波器实现的详细信息，请参阅 EncFilt 手册页面。

主编码器上的滤波器（由 EncFilt 控制）和辅助编码器上的滤波器（由 AuxEncFilt 控制）是两个独立的滤波器。然而，它们的实现方法是相同的。

AuxEncDir（辅助编码器方向反转）

项 目	内容
关键字（Mnemonic）	AuxEncDir
CAN 编码	61
类型	参数（Parameter）
访问权限	只读（No）
允许在运动中	否（No）
允许在电机开启时	否（No）
是否存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	0
最大值	1
默认值	0（表示默认无反向）

启用辅助编码器方向的反向读取（适用于所有类型的编码器）。

如果 AuxEncDir == 0，辅助编码器将按连接到驱动器的方式读取。

如果 AuxEncDir == 1，辅助编码器将以反向方向读取。

项 目	内容
关键字（Mnemonic）	PDEncFilt
CAN 编码	62
类型	参数（Parameter）
访问权限	读/写（Read / Write）
轴相关	是（Yes）
数值类型	32 位整数（’int’ 32 bit）
用户单位	无（No user units）
允许在运动中	否（No）
允许在电机开启时	是（Yes）
存储到闪存	是（Yes）
最小值	-2,147,483,648（整型最大负值）
最大值	2,147,483,647（整型最大正值）
默认值	0

定义：
该参数用于控制位置反馈信号（通常是编码器位置反馈）是否经过数字滤波处理，以及滤波的具体参数。

常规用途：
调节 PD 反馈的数字滤波器参数，优化系统对高频噪声的抑制能力，从而获得更平滑和稳定的定位反馈。

使用建议：

- 在实际应用中，一般建议从默认值 0 开始，如需要减少噪声可以设置为正整数（越大滤波越强），但应在确保系统响应速度足够的前提下调整。
- 设置过大可能导致反馈延迟，影响动态响应。

项 目	内容
关键字（Mnemonic）	PDEncDir
CAN 编码	63
类型	参数（Parameter）
访问权限	读/写（Read / Write）
轴相关	是（Yes）
数值类型	32 位整数（’int’ 32 bit）
用户单位	无（No user units）
允许在运动中	否（No）
允许在电机开启时	是（Yes）
存储到闪存	是（Yes）
最小值	-2,147,483,648（整型最大负值）
最大值	2,147,483,647（整型最大正值）
默认值	0

定义：
此参数用于控制位置反馈信号（一般由编码器提供）是否反向。设置为非零值时，即为反转反馈信号的方向，有助于匹配实际机械运动或硬件接线的反向情况。

应用场景：

- 配合实际硬件接线，确保反馈方向与运动控制逻辑一致。
- 在系统调试或安装硬件时调整，保证位置反馈的方向正确。

设置建议：

- 默认值为 0（无反转）。
- 如果系统实际反馈方向与控制逻辑相反，可以将值设置为 1 进行反转。

注意事项：

- 在运动过程中不能修改此参数，建议在初始化或停机状态调整。

项 目	内容
关键字（Mnemonic）	UsrUnits
CAN 编码	64
类型	参数（Parameter）
访问权限	读/写（Read / Write）
允许在运动中	否（No）
允许在电机开启时	否（No）
存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	1
最大值	2,147,483,647（32 位最大整数值）
默认值	65,536

UsrUnits 允许用户以编码器计数以外的单位读取位置及其导数。 UsrUnits 是所需单位与编码器计数之间的比率。关于用户单位如何工作的完整说明可以在本文档的开头找到。

示例：

如果用户希望以毫米为单位查看位置读数，并且每 5 个编码器计数相当于 1 毫米，则将 UsrUnits 设置为 5。

现在，位置读数将以毫米为单位接收，速度以毫米/秒为单位，且加速度以毫米/秒 ^{2} 为单位。

另请参阅：

AuxUsrUnits , PDUsrUnits

项 目	内容
关键字（Mnemonic）	AuxUsrUnits
CAN 编码	65
类型	参数（Parameter）
访问权限	读写（Read / Write）
允许在运动中	否（No）
允许在电机开启时	否（No）
是否存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	1
最大值	2,147,483,647
默认值	65,536
用户单位	无用户定义单位（No user units）
实现状态	已实现（Implemented）
示例	如果用户希望辅助位置以毫米（mm）表示，并且每 5 个编码器计数对应 1mm，可以将 UsrUnits 设置为 5。此时，辅助位置会以毫米显示，速度以 mm/sec，加速度以 mm/sec² 显示。
关联参数	UsrUnits, PDUsrUnits

说明

辅助用户单位（AuxUsrUnits）允许用户以非编码器计数的单位读取辅助反馈位置及其导数。AuxUsrUnits 是所需单位与编码器计数之间的比率。关于用户单位如何工作的完整说明可以在本文档的开头找到。

示例：

如果用户希望以毫米为单位查看辅助位置读数，并且辅助编码器上的每 5 个编码器计数相当于 1 毫米，则将 UsrUnits 设置为 5。

辅助位置读数现在将以毫米为单位接收，速度以毫米/秒为单位，且加速度以毫米/秒²为单位。

另请参阅：

UsrUnits, PDUsrUnit

PDUsrUnits（脉冲方向命令单位）

项 目	内容
关键字（Mnemonics）	PDUsrUnits
CAN 编码	66
类型	参数（Parameter）
访问权限	读写（Read / Write）
允许在运动中	否（No）
允许在电机开启时	否（No）
是否存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	1
最大值	2,147,483,647
默认值	65,536
用户单位	无用户定义单位（No user units）
实现状态	已实现（Implemented）

说明

PDUsrUnits 允许用户以非脉冲单位读取脉冲方向命令及其衍生物。 PDUsrUnits 是期望单位与脉冲之间的比率。关于用户单位如何工作的完整解释可以在本文档的开头找到。

示例：

如果用户希望以毫米为单位查看位置命令，并且每 5 个脉冲相当于 1 毫米，则将 UsrUnits 设置为 5。

现在位置命令将以毫米为单位接收。

另请参阅：

AuxUsrUnits , UsrUnits

项 目	内容
关键字（Mnemonic）	EmulRat
CAN 编码	69
类型	参数（Parameter）
访问权限	读写（Read / Write）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
是否存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	0
最大值	65,536
默认值	0
用户单位	无用户定义单位（No user units）
实现状态	已实现（Implemented）

EmulRat 确定主编码器输入与编码器仿真输出之间的比率。

有 4 种类型的仿真编码器信号：

1. A – 数字增量编码器信号 A 侧的逻辑信号
2. B – 数字增量编码器信号 B 侧的逻辑信号
3. Z – 索引信号
4. Pulse – 对于仿真编码器信号的每个边缘转换，生成一个 5 微秒的脉冲

仿真编码器信号的输出仅路由到差分输出信号，如下所示：
- 仿真编码器 A 连接到差分输出 1
- 仿真编码器 B 连接到差分输出 2
- 仿真编码器 Z 连接到差分输出 3
- 仿真编码器脉冲 连接到差分输出 4

它们可以通过选择值为 1 的特定输出的 DOutSelect 进行配置。

示例：

如果 EmulRat = 8，则每 8 个输入脉冲将生成一个仿真编码器输出脉冲。

关联参数

- DOutPort：控制差分输出端口
- DOutSelect：选择输出对应的模拟信号类型

项 目	内容
关键字（Mnemonic）	ModRev
CAN 编码	70
类型	参数（Parameter）
访问权限	读写（Read / Write）
允许在运动中	否（No）
允许在电机开启时	否（No）
是否存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	0
最大值	2,000,000,000
默认值	0（关闭模余）
用户单位	以用户设定单位（In user units）

说明

ModRev 用于控制器的模模式。在模模式下，主位置（Pos）在 0 和 ModRev 之间。当实际位置超过 ModRev 时，Pos 的值被计算为 ModRev 的一部分。这允许旋转轴在同一方向上无限移动而不超过任何数值限制。

ModRev = 0 关闭模模式。

注意：

1. 如果 ModRev 不为零，和/或不够高，自动相位（电机电气周期与控制器对齐的上电过程）可能会失败。请在执行自动相位之前将 ModRev 设置为零，并在自动相位完成后将其设置为所需值。

1. 不要将模模式与输入整形一起使用。

1. 如果您使用 SetPosition 手动设置位置为超过 ModRev 范围的值，可能会发生意外行为。

1. 无限运动（ModRev 不为 0）可以与平滑（Jerk 不为 0）一起使用，但用户必须根据以下指南正确设置 ModRev 和 Jerk：

    1. 在最大速度下，模数（移动 ModRev 距离）的一次完整旋转的时间（以样本数为单位）应长于与 Jerk 值相关的时间（2^{Jerk} 是样本数）。

这可以通过正确设置 ModRev（足够大）轻松解决。

例如，如果最大速度为 10,000,000 [counts/sec]，Jerk = 9（即 2^9 = 512 样本），控制器采样率为 16,384 [samples/sec]，则 ModRev 必须大于 10,000,000 / 16,384 * 512 = 312,500 [counts]。

1. 与 Jerk 相关的样本数（为 2^{Jerk} [样本]）和 ModRev（以 [counts] 为单位）的乘积必须小于 (2^{31} -1)。

   这可以通过正确设置 ModRev（足够小）轻松解决。

   继续上述示例，如果 Jerk = 9，则 ModRev 必须小于：(2^{31} -1) / 512 = 4,194,304 [counts]

如果由于某种原因，您无法根据上述指南将 ModRev 设置为所需的高或低值，请咨询 Agito 以获取正确的设置方法。

相关参数

- **Jerk**：运动平滑参数

项 目	内容
关键字（Mnemonic）	MotorOn
CAN 编码	130
类型	参数（Parameter）
访问权限	读写（Read / Write）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
是否存储到闪存	否（No）
轴相关	是（Yes）
最小值	0
最大值	1
默认值	0（电机关闭）
用户单位	无（No user units）
实现状态	已实现（Implemented）

说明：

MotorOn：启用/禁用电机。

MotorOn = 0 禁用电机。

MotorOn = 1 启用电机。

当电机被禁用时，电源不会施加到电机上，并且不受驱动器控制。

电机可以由于故障由驱动器内部禁用（参见 ConFlt）。

当用户重新启用电机时，ConFlt 被清除。如果导致故障的实际状态未得到纠正（例如：过温导致电机禁用，用户在冷却前尝试启用电机），电机将保持禁用状态。

某些命令和参数更改仅在电机禁用时允许。请参阅每个参数的属性。

关键字（Mnemonic）	InjectType
CAN 编码	112
类型	参数（Parameter）
访问权限	读写（Read / Write）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	0
最大值	6
默认值	0
用户单位	无（No user units）
实现状态	已实现（Implemented）

InjectType 决定在使用注入功能时将注入哪种类型的信号。可能的注入类型有：

值	注入类型
0	无注入
1	直接正弦信号
2	添加正弦信号
3	直接方波信号
4	添加方波信号
5	直接脉冲

所需的信号类型被注入到由 InjectPoint 选择的控制回路中。

直接正弦信号

注入一个正弦信号代替任何其他参考信号。例如，如果注入信号被注入到位置回路中，生成的位置信号将被忽略并替换为正弦参考。

添加正弦信号

添加一个正弦信号到任何其他参考信号。例如，如果注入信号被注入到位置回路中，正弦信号将被添加到生成的位置信号中。

直接方波信号

注入一个方波信号代替任何其他参考信号。例如，如果注入信号被注入到位置回路中，生成的位置信号将被忽略并替换为方波参考。这可以用作模拟阶跃响应。

添加方波信号

添加一个方波信号到任何其他参考信号。例如，如果注入信号被注入到位置回路中，方波信号将被添加到生成的位置信号中。

直接脉冲

注入一个单一脉冲代替任何其他参考信号。

另请参阅：

InjectPoint , InjectCurrAmp , InjectVelAmp , InjectPosAmp , InjectFreq , InjectValue , InjectTimeOn , InjectCurrDC

ScheduleSet（调度集）

项目	内容
关键字（Mnemonic）	ScheduleSet
CAN 编码	261
类型	参数（Parameter）
访问权限	读写（Read / Write）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	1
最大值	5
默认值	1
用户单位	无（No user units）
实现状态	已实现（Implemented）

补充说明：

- `ScheduleSet` 主要用于配置 **Gain Scheduling**（增益调度）功能的参数集。
- 详细的调度参数和行为定义，请参考 `ScheduleMode` 关键字页面。
- 调度可以动态调整控制参数，以适应不同工况，提高系统性能。

示例：

- 设置 `ScheduleSet = 3` 表示使用第 **3** 号调度方案，应用特定的增益配置。

MotionSamples（运动样本时间数组）

项目	内容
关键字（Mnemonic）	MotionSamples
CAN 编码	267
类型	只读参数（Read only）
访问权限	允许运动中读取（Yes）
允许在电机开启时	允许（Yes）
数组索引范围	1 ： 4
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	-1
最大值	2,147,483,647（32 位最大值）
默认值	-1（表示尚未进行任何运动）
用户单位	无（No user units）
实现状态	已实现（Implemented）

说明：

Index 1: 样本分析器

Index 2: 样本直至目标

Index 3: 样本直至达到目标

Index 4: 样本稳定时间

MotionSamples[] 是一个数组参数，用于报告最近完成的运动的时间持续时间。

时间以控制器样本数报告（通常对于 Agito 的控制器：1/16384=61μs 每个样本）。

报告三个时间，如下所示：

MotionSamples[1]: 运动轮廓本身的时间。

与电机/轴的实际运动无关。

MotionSamples[2]: 运动轮廓本身的时间加上电机稳定到目标所需的时间（不包括 InTargetTime）。

MotionSamples[3]: 运动轮廓本身的时间加上电机稳定到目标所需的时间（包括 InTargetTime）。

上电或重置后，所有 MotionSamples[] 元素都设置为 -1，以指示尚未执行任何运动。

另请参阅： InTargetTime, InTargetTol, InTargetStat.

PosErr（位置误差）

项目	内容
关键字（Mnemonic）	PosErr
CAN 编码	18
类型	只读参数（Read only）
访问权限	允许运动中读取（Yes）
允许在电机开启时	允许（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	-2,147,483,648（32 位有符号整数的最小值）
最大值	2,147,483,647（32 位有符号整数的最大值）
默认值	0
用户单位	以用户单位（In user units）
实现状态	已实现（Implemented）

说明：

PosErr 返回位置误差的值，以用户单位 (UsrUnits) 表示。位置误差是位置命令 (PosRef) 与实际位置之间的差异。如果 PosErr 的值超过 MaxPosErr 中定义的最大允许位置误差，电机将被禁用。

示例：

如果 PosRef = 1000 且 Pos = 990，则 PosErr = 10

另请参阅：

N/A

VelErr（速度误差）

项目	内容
关键字（Mnemonic）	VelErr
CAN 编码	19
类型	只读参数（Read only）
访问权限	允许运动中读取（Yes）
允许在电机开启时	允许（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	-2,147,483,648（32 位有符号整数的极小值）
最大值	2,147,483,647（32 位有符号整数的极大值）
默认值	0
用户单位	以用户单位/秒（In user units/sec）
实现状态	已实现（Implemented）

说明：

VelErr 返回速度误差的值，单位为用户单位 (UsrUnits)/秒。速度误差是速度命令 (VelRef) 和实际速度 (Vel) 之间的差值。如果 VelErr 的值超过 MaxVel Err 中定义的最大允许位置误差，电机将被禁用。

示例：

如果 VelRef = 10000 且 Vel = 9900 则 VelErr = 100

另请参阅：

UsrUnits , MaxVel Err

DPosRef（目标位置的偏差）

项目	内容
关键字（Mnemonic）	DPosRef
CAN 编码	155
类型	只读参数（Read only）
访问权限	允许运动中读取（Yes）
允许在电机开启时	允许（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	-2,147,483,648（32 位有符号整数极小值）
最大值	2,147,483,647（32 位有符号整数极大值）
默认值	0
用户单位	以用户单位（In user units）
实现状态	已实现（Implemented）

说明：

- DPosRef 表示目标位置与当前位置（Pos）的偏差值，单位为用户单位。
- 通常用于反馈调节或控制算法中的偏差检测。

ComtStatus（换向状态）

项目	描述
关键字（Mnemonic）	ComtStatus
CAN 编码	143
类型	参数（Parameter）
数组索引范围	1 ： 2
访问权限	只读（Read only）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
最小值	0
最大值	2,147,483,647
默认值	0

状态值含义（索引 1）

值	状态描述
0	尚未完成（等待或未启动自动配合）
1	正在处理中（自动配合中）
100	成功完成（自动配合成功）
200	不需要自动配合（无需自动配合）
300	成功完成（使用霍尔传感器）
400	成功完成（使用霍尔传感器）
500	成功完成（参数已修改！！）
600	老化模式下的虚拟换向
-1	失败；电机意外下使能（参见 MotorReason 指令）
-2	失败；未知的自动换相方式
-3	失败；搜索超过 360 度

值	状态描述
-4	失败；检测到编码器断开连接
-5	需要换向；相关参数被修改
-6	失败；硬件未准备好。断电重启
-7	失败；霍尔信号值非法。断电重启
-8	需要换向；Cenreal-i 远程单元断开连接
-9	失败；自学习时遇到非法霍尔值（0 或 7）
-10	失败；自学习时遇到非法霍尔序列
-11	失败；非法霍尔值。EncDir 被修改
-12	失败；非法霍尔序列。EncDir 被修改

Ia（相“A”电流）

项目	内容
关键字（Mnemonic）	Ia
CAN 编码	9
类型	参数（Parameter）
访问权限	只读（Read only）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	-2,147,483,648（32 位有符号最小值）
最大值	2,147,483,647（32 位有符号最大值）
默认值	0
用户单位	无（No user units）

说明：

Ia 表示相“A”线的电流值，单位为毫安（mA），根据硬件连接方案定义。

示例：
若检测到的电流为 1500 mA，则 Ia 的值为 1500。

该参数常用于监控每相电流的实时值，以确保电机工作在安全范围内。

如果需要其他相电流参数（如 Ib, Ic, Id）

Ib（相“B”电流）

项目	内容
关键字（Mnemonic）	Ib
CAN 编码	10
类型	参数（Parameter）
访问权限	只读（Read only）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	-2,147,483,648（32 位有符号最小值）
最大值	2,147,483,647（32 位有符号最大值）
默认值	0
用户单位	无（No user units）

说明：

Ib 表示相“B”线的电流值，单位为毫安（mA），依据硬件连接配置确定。

示例：
若检测到的电流为 1600 mA，则 Ib 的值为 1600。

此参数用于监控相“B”线的电流状态，有助于检测异常或优化控制。

Id（直轴电流）

项目	内容
关键字（Mnemonic）	Id
CAN 编码	11
类型	参数（Parameter）
访问权限	只读（Read only）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	-2,147,483,648（32 位有符号最小值）
最大值	2,147,483,647（32 位有符号最大值）
默认值	0
用户单位	无（No user units）

说明：

I 在矢量控制模式下（参见 ControlMode 关键字），电流控制回路在 Id 和 Iq 上闭合。 Iq 闭环以 CurrRef 作为其参考，而 Id 闭环以零作为其参考。

Id 是从 Ia 和 Ib 计算得出的，是 非 有效电流，意味着不产生扭矩的电流（与产生扭矩的 Iq 相对）。

Id 以[mA]为单位。

请注意，即使控制器未使用矢量控制，Id 也会被计算。用户可以监控和分析 Id 和 Iq 以估计电机运行的效率。

通常 Iq 应等于 CurrRef，而 Id 应等于零。

注意：

在旧的固件版本（早于版本 1.3.0）中，Id 和 Iq 是交换的。从固件版本 1.3.0 开始，这一问题已修复，并符合上述文档。

另请参阅：IdRef, IdErr, IqRef, Iq, IqErr, Control Mode。

Iq（有效电流/转矩电流）

项目	内容
关键字（Mnemonic）	Iq
CAN 编码	12
类型	参数（Parameter）
访问权限	只读（Read only）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	-2,147,483,648（32 位有符号最小值）
最大值	2,147,483,647（32 位有符号最大值）
默认值	0
用户单位	无（No user units）
实现版本	固件版本 1.3.0 及以上

说明：

在矢量控制模式下（参见 ControlMode 关键字），电流控制回路在 Id 和 Iq 上闭合。Iq 闭环以 CurrRef 为参考，Id 闭环以零为参考。

Iq 是从 Ia 和 Ib 计算得出的，是有效电流，意味着产生扭矩的电流（与 Id 不产生扭矩相反）。

Iq 以[mA]为单位。

请注意，即使控制器未使用矢量控制，Iq 也会被计算。用户可以监控和分析 Id 和 Iq 以估计电机运行的效率。

通常 Iq 应等于 CurrRef，Id 应等于零。

注意：

在旧的固件版本（早于版本 1.3.0）中，Id 和 Iq 是交换的。从固件版本 1.3.0 开始，这一问题已修复，并符合上述文档。

另请参阅：IdRef, Id, IdErr, IqRef, IqErr, 控制模式。

Va（相“A”电压）

项目	内容
关键字（Mnemonic）	Va
CAN 编码	13
类型	参数（Parameter）
访问权限	只读（Read only）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	动态（依赖于遥控单元，随设备不同而变化）
最大值	动态（依赖于遥控单元，随设备不同而变化）
默认值	动态（依赖于遥控单元）

说明：

- Va 表示相“A”线上的电压，单位为全 PWM 周期的百分比（%）。
- **“全 PWM”**指最大调制幅值，Va 值范围随硬件远程单元不同而变化，动态确定。
- **用途：** 实时反映相“A”的电压水平，有助于监视驱动电压状态。

Va 返回施加到相位“A”的电压值，以 0.1% 的全 PWM 百分比表示。

示例： Va 的值为 53，意味着对相位 A 的 PWM 命令为 5.3%（该值以 0.1%为单位给出）。

与 Va 相关的参数（如 Vb, Vc）

Vb（相“B”电压）

项目	内容
关键字（Mnemonic）	Vb
CAN 编码	14
类型	参数（Parameter）
访问权限	只读（Read only）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	动态（依赖遥控单元，随设备变化）
最大值	动态（依赖遥控单元，随设备变化）
默认值	动态（依赖遥控单元）

说明：

- Vb 表示相“B”线的电压值，单位为全 PWM 周期中的百分比（%）。
- **“全 PWM”**代表最大调制幅值，Vb 的范围随硬件遥控单元的不同而变化，动态确定。
- **用途**：用以实时监测相“B”线的电压状态，有助于诊断和调试。

Vc（相“C”电压）

项目	内容
关键字（Mnemonic）	Vc
CAN 编码	15
类型	参数（Parameter）
访问权限	只读（Read only）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	动态（依赖远程单元，随硬件变化）
最大值	动态（依赖远程单元，随硬件变化）
默认值	动态（依赖远程单元）

说明：

- Vc 表示相“C”线的电压值，百分比表示，范围为相应全 PWM 幅值。
- **“全 PWM”**指最大调制幅值，范围随硬件不同而变化，动态确定。
- **用途**：用于实时监测相“C”线的电压状态，便于调试和状态监控。

Vd（直轴电压）

项目	内容
关键字（Mnemonic）	Vd
CAN 编码	16
类型	参数（Parameter）
访问权限	只读（Read only）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	动态（依赖远程单元，随硬件变化）
最大值	动态（依赖远程单元，随硬件变化）
默认值	动态（依赖远程单元）

说明：

Vd 参数仅在矢量控制激活时使用（请参阅 ControlMode 关键字）。

Vd 是 Id 电流控制回路中 PI 控制器的输出。

Vd 和 Vq 用于计算 Va 和 Vb（所需的电机相电压）。

Vd 以内部单位表示（通常在 Agito 的控制器中：Vd 的值为 4577 时表示 100% PWM）。

当矢量控制未激活时，Vd 等于零。

另请参阅：Vq, IdRef, IqRef, Id, Iq, IdErr, IqErr, ControlMode, Va, Vb, Vc。

Vq（交轴电压）

项目	内容
关键字（Mnemonic）	Vq
CAN 编码	17
类型	参数（Parameter）
访问权限	只读（Read only）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	动态（依赖远程单元，随硬件变化）
最大值	动态（依赖远程单元，随硬件变化）
默认值	动态（依赖远程单元）

说明：

Vq 参数仅在矢量控制激活时使用（请参阅 ControlMode 关键字）。

Vq 是 PI 控制器在 Iq 电流控制回路中的输出。

Vd 和 Vq 用于计算 Va 和 Vb（所需的电机相电压）。

Vq 以内部单位表示（通常在 Agito 控制器中：Vq 值为 4577 时表示 100% PWM）。

当矢量控制未激活时，Vq 等于零。

另请参阅：Vd, IdRef, IqRef, Id, Iq, IdErr, IqErr, ControlMode, Va, Vb, Vc。

HomingStat（回零状态）

项目	内容
关键字（Mnemonic）	HomingStat
CAN 编码	342
类型	参数（Parameter）
访问权限	只读（Read only）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	-2,147,483,648（32 位有符号最小值）
最大值	2,147,483,647（32 位有符号最大值）
默认值	0
用户单位	无（No user units）

说明：

- HomingStat 表示当前回零（Homing）过程的状态或代码。
- 详细的回零功能描述可参考 HomingOn 关键词页面，涉及回零流程、状态机和控制参数。
- **用途**：监控回零操作的实时状态，判断是否完成或是否出现异常。

值	描述
0	未执行
100	成功完成
-1	由于参数错误而失败
-2	由于超时而失败
-3	因电机意外下使能（参考 MotorReason 指令）导致失败
-4	由于运动　　误而失败
-5	由于步骤类型错误而失败
-6	由于运动中而失败
-7	由于步骤数量超出范围

值	描述
-8	由于意外限制（障碍等）导致失败
-9	无法设置位置（参见 SetPosition 指令）
-10	运动/齿轮模式超出范围
-11	误差补偿值超出范围
-12	自动换相未执行，操作失败

项 目	内容
关键字（Mnemonic）	MotionReason
CAN 编码	43
类型	只读参数（Read only）
访问权限	只读（Read only）
轴相关	是（Yes）
数值类型	32 位整数（’int’ 32 bit）
用户单位	无（No user units）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
值范围	0 到 2,147,483,647
默认值	0

功能说明

MotionReason 返回一个值，表示导致最近一次运动结束的原因。Begin 命令将 MotionReason 重置为 0。

下表列出了 MotionReason 的可能值及其含义：

- Value = 0: 正常
- Value = 1: 停止请求
- Value = 2: 中止请求
- Value = 3: 停止往复请求
- Value = 4: 负向限位开关
- Value = 5: 正向限位开关
- Value = 6: 负向位置限制
- Value = 7: 正向位置限制
- Value = 8: 电机意外下使能（参见 MotorReason 指令）
- Value = 9: 用户请求停止 ECAM
- Value = 10: 用户请求停止 FIFO
- Value = 11: 参考零位检测
- Value = 12: 用户请求停止 CNCA
- Value = 13: 回零超时

- Value = 14: 切换到电流模式
- Value = 15: 回零到机械限位
- Value = 16: 零点开关
- Value = 17: 切换到力控模式
- Value = 18: CNC 参与轴下使能
- Value = 19: CNC 的参与轴被停止
- Value = 20: CNC 的参与轴被中止
- Value = 21: 通过开关量输入停止
- Value = 22: 通过开关量输入中止
- Value = 23: CNCA 参与轴触发正向限位开关/负向限位开关
- Value = 24: CNCA 参与轴到达正向位置限制/负向位置限制
- Value = 25: 用户请求停止 CNCB
- Value = 26: CNCB 参与轴触
- Value = 27: CNCB 参与轴触发正向限位开关/负向限位开关
- Value = 28: 数字量输入受控停止
- Value = 29: 用户请求停止矢量运动
- Value = 30: 矢量参与轴下使能
- Value = 31: 矢量参与轴被停止
- Value = 32: 矢量参与轴被中止
- Value = 33: 矢量参与轴触发正向限位开关/负向限位开关
- Value = 34: 矢量参与轴到达正向位置限制/负向位置限制
- Value = 35: 用户请求停止插补运动
- Value = 36: 插补参与轴下使能
- Value = 37: 插补参与轴被停止
- Value = 38: 插补参与轴被终止
- Value = 39: 插补参与轴触发正向限位开关/负向限位开关
- Value = 40: 插补参与轴到达正向位置限制/负向位置限制
- Value = 41: 点动运动超过负向或正向位置限制而停止

示例：

如果运动因中止命令而结束，但在减速过程中超过了正向软件限位，然后遇到了限位开关，MotionReason 将具有值 2，表示停止的原始原因，并忽略任何可能导致运动停止的后续事件。

-

项	内容
关键字（Mnemonic）	InTargetStat
CAN 编码	268
类型	只读参数（Read only）
访问权限	只读（Read only）
是否与轴相关	是（Yes）
数值类型	32 位整数（'int' 32 bit）
用户单位	无（No user units）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
值范围	0 到 4
默认值	0

状态码与含义

值	状态描述	说明
0	电机关闭	运动系统未启动或电机未激活
1	伺服使能	伺服已启用，准备或正在运动
2	运动中	当前正在执行运动
3	等待 InTargetTime 结束	已达到目标位置，但还在等待确认时间（InTargetTime）完成以确认目标到达
4	目标已达到	在允许误差（InTargetTol）范围内持续时间超过 InTargetTime 后确认目标已到达

补充说明

工作原理：
当位置误差在用户单位中小于 `InTargetTol`，并且持续时间达到 `InTargetTime` 毫秒，`InTargetStat` 会被设置为 **4**，表示目标已到达。

相关参数：

- **InTargetTol**：位置误差阈值
- **InTargetTime**：持续确认的时间

MotionMode（运动模式）

项	内容
关键字（Mnemonic）	MotionMode
CAN 编码	141
类型	读/写参数（Read / Write）
是否与轴相关	是（Yes）
数值类型	32 位整数（'int' 32 bit）
用户单位	无（No user units）
允许在运动中	否（No）（运动中不能更改）
允许在电机开启时	是（Yes）
存储到闪存	是（Yes）
范围	-1 到 19
默认值	-1（表示未设置或无效）

说明

MotionMode 确定将在下一个 Begin 命令后执行的运动类型。在当前运动结束之前，MotionMode 不能更改。

MotionMode 的值及相关的运动类型为：

MotionMode	结果运动
0	Jog – 电机将达到 Speed 中设置的速度，并在接收到 Stop 命令之前以恒定速度继续。运动使用的加速和减速由 Accel 和 Decel 定义。运动也可以根据 Jerk 的值进行平滑。
1	PTP – 移动到由 RelTrgt 或 AbsTrgt（如果 RelTrgt = 0）定义的位置。运动的其他参数由 Accel、Speed、Decel 和 Jerk 定义。
2	PTP 重复 – 从当前位置到由 RelTrgt 或 AbsTrgt（如果 RelTrgt = 0）定义的位置的重复运动，然后返回。运动的其他参数由 Accel、Speed、Decel 和 Jerk 定义。 电机将在运动的每一端等待由 RptWait 确定的时间。运动使用 StopRep 停止。
3	脉冲方向直接模式 – 位置环的参考位置直接从脉冲方向输入接收。
4	脉冲方向间接模式 – 此运动的位置信号根据脉冲方向输入确定。然而，接收到的脉冲数不会立即用作位置环的参考。 接收到的脉冲数被添加到当前位置目标。分析器使用所有运动参数的值构建到新目标的运动轨迹：Accel、Decel、Speed、Jerk。 这意味着即使一次输入大量脉冲，结果运动也不会像阶跃响应，而是更平滑。请注意，对快速输入的响应将更平滑，但也更慢。

5	齿轮直接运动
6	齿轮间接运动
7	ECAM 直接运动
8	ECAM 间接运动
9	FIFO 运动
10	从动位置参考
11	CNC 组 A 运动
12	摇杆直接运动，输入模拟信号表示位置参考
13	摇杆间接运动，输入模拟信号表示位置参考，此分析器可由用户设置
14	摇杆直接运动，输入模拟信号表示速度参考
15	摇杆间接运动，输入模拟信号表示速度参考，此分析器可由用户设置
16	矢量运动
17	CNC 组 B 运动
18	样条缓冲运动

**注意：某些运动模式在固件升级到某个版本时可用。

另请参阅：

Accel, Decel, Speed, Jerk, AbsTrgt, RelTrgt, Begin, RptWait, StopRep

Jerk（平滑度参数）

项	内容
关键字（Mnemonic）	Jerk
CAN 编码	139
类型	读/写参数（Read / Write）
是否与轴相关	是（Yes）
数值类型	32 位整数（’int’ 32 bit）
用户单位	无（No user units）
允许在运动中	否（No）——运动过程中不能修改
允许在电机开启时	是（Yes）
存储到闪存	是（Yes）
范围	0 到 13
默认值	0

Jerk 定义了在所有基于轮廓的运动中建立所需加速度（或减速度）所需的时间。

建立加速度的时间通过以下公式计算：

Jerk Time = SAMPLE_TIME * 2^{Jerk}

SAMPLE_TIME 是控制器的采样时间，通常为 1/16384=61 微秒，适用于 Agito 控制器。

例如，如果 JERK = 7，则 Jerk 时间为 7.808 毫秒。

请注意，Jerk 是唯一一个在运动过程中无法修改的运动轮廓参数。

Jerk Time 在加速开始或结束时使用，减速时也类似。

显然，随着 Jerk 值的增加，运动时间更长。然而，运动更平滑，预期的超调减少。应根据每个特定应用程序调整 Jerk 以获得最佳性能。

注意：

1. 平滑（Jerk 不为 0）可以与无限运动（ModRev 不为 0）一起使用，但用户必须根据以下指南正确设置 Jerk 和 ModRev：

    1. 在最大速度下，模数的完整旋转时间（以样本数为单位）应长于与 Jerk 值相关的时间（2^{Jerk}是样本数）。

        这可以通过正确设置 ModRev（足够大）轻松解决。

例如，如果最大速度为 10,000,000 [counts/sec]且 Jerk = 9（即 2^9= 512 样本），控制器采样率为 16,384 [samples/sec]，则 ModRev 必须大于 10,000,000 / 16,384 * 512 = 312,500 [counts]。

1. 与 Jerk 相关的样本数（为 2^{Jerk}[样本]）和 ModRev（以[counts]为单位）的乘积必须小于(2^{31}-1)。

   这可以通过正确设置 ModRev（足够小）轻松解决。

   继续上述示例，如果 Jerk = 9，则 ModRev 必须小于：

   (2^{31}-1) / 512 = 4,194,304 [counts]

请咨询 Agito 以获得设置 ModRev 和 Jerk 的正确方法，以满足需要平滑和无限运动的特定应用程序，如果由于某种原因，您无法根据上述指南将 ModRev 设置得足够高或足够低。

1. Jerk（平滑）值出现在 PCSuite 的几乎所有运动窗口中。然而，请注意，当按下运动按钮时，Jerk 值不会写入控制器（与加速度和速度等其他参数不同）。这是因为在伺服使能时无法修改 Jerk。因此，如果需要新的平滑（Jerk）值，用户应更改平滑值，然后禁用电机，使用“应用更改”写入新的 Jerk 值（现在将写入，因为电机已禁用），然后使用运动按钮（运动现在将使用新的平滑/Jerk 值）。

另请参阅：

Accel, Decel, Speed, RelTrgt, AbsTrgt 和 ModRev。

Accel（加速度）

项	内容
关键字（Mnemonic）	Accel
CAN 编码	136
类型	读/写参数（Read / Write）
是否与轴相关	是（Yes）
数值类型	32 位整数（'int' 32 bit）
用户单位	用户单位（User units）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	是（Yes）
范围	100 到 2,000,000,000
默认值	100,000

Accel 是用于所有运动的加速度，单位为用户单位/秒²。当应用间接模式时，该加速度也用于脉冲-方向命令输入。

Accel 可以在运动过程中更改，最新值将立即生效。如果当前运动仍在加速中，加速度率将会改变。如果达到恒定速度，则在下次需要加速时将使用新的加速度值。

另请参阅：

AbsPosTrgt, RelPosTrgt, Vel, Speed, Decel, MaxAcc

Decel（减速度）

项	内容
关键字（Mnemonic）	Decel
CAN 编码	137
类型	读/写参数（Read / Write）
是否与轴相关	是（Yes）
数值类型	32 位整数（'int' 32 bit）
用户单位	用户单位（User units）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	是（Yes）
范围	100 到 2,000,000,000（用户单位/秒²）
默认值	100,000

定义和功能说明

Decel 是用于所有运动的减速度，单位为用户单位/秒²。当应用间接模式时，此减速度也用于脉冲-方向命令输入。

Decel 可以在运动过程中更改，最新值将立即生效。如果当前运动仍在减速，减速度将被更改。否则，下次需要加速时将使用新的加速度值。

Decel 也用于在输入停止命令后进行减速。

对于紧急停止，使用 EmrgDec 。

另请参阅：

EmrgDec ， Stop

EmrgDecel（紧急减速度）

项	内容
关键字（Mnemonic）	EmrgDec
CAN 编码	140
类型	读/写参数（Read / Write）
是否与轴相关	是（Yes）
数值类型	32 位整数（’int’ 32 bit）
用户单位	用户单位（User units /秒²）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	是（Yes）
范围	100 到 2,000,000,000（用户单位/秒²）
默认值	100,000

EmergDec 是 UsrUnits /Sec²中的紧急减速。此参数通常应设置为比 decal 高得多的减速度。EmrgDec 的值在以下原因之一需要停止运动时使用：

- 硬件限制
- 软件限制

另请参阅：

RevPlim , FwdPlim , Decel , DInMode

Speed（速度）

项	内容
关键字（Mnemonic）	Speed
CAN 编码	138
类型	读/写参数（Read / Write）
是否与轴相关	是（Yes）
数值类型	32 位整数（'int' 32 bit）
用户单位	用户单位每秒（UsrUnits/Sec）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	是（Yes）
范围	-1,300,000,000 到 1,300,000,000 用户单位/秒（-1.3 亿到 1.3 亿）
默认值	0

定义与作用

Speed 的值决定了运动的期望速度，以 UsrUnits/Sec 为单位。 Speed 可以在运动过程中动态更改。 如果当前运动处于加速阶段或恒速阶段，它将更改为更新后的速度。

示例：

Speed = 10,000

...（所有其他运动参数和模式）

开始

上述序列以期望速度 10,000 启动运动。

如果在实际速度达到 10,000 后 Speed 被更改：

Speed = 20,000

（无需重新开始）

电机将使用当前设置的加速度加速到 20,000。

另请参阅：

Accel , Decel , AbsTrgt , RelTrgt , Begin

MaxVel（最大速度）

项	内容
关键字（Mnemonic）	MaxVel
CAN 编码	80
类型	读/写参数（Read / Write）
是否与轴相关	是（Yes）
数值类型	32 位整数（'int' 32 bit）
用户单位	用户单位每秒（UsrUnits/Sec）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	是（Yes）
范围	0 到 1,300,000,000 用户单位/秒
默认值	100,000

定义与作用

MaxVel 确定受控电机的最大允许速度。速度环的速度参考受 MaxVel 限制。如果反馈的实际读数超过 MaxVel 的 25%，电机将被禁用。

MaxAcc（最大加速度）

项	内容
关键字（Mnemonic）	MaxAcc
CAN 编码	81
类型	读/写参数（Read / Write）
是否与轴相关	是（Yes）
数值类型	32 位整数（’int’ 32 bit）
用户单位	无专用用户单位（No user units）
允许在运动中	否（No）——仅在运动前设置，不在运动过程中动态变化
允许在电机开启时	是（Yes）
存储到闪存	是（Yes）
范围	-2,147,483,648 到 2,147,483,647（整型全部范围）
默认值	0

定义与作用

MaxAcc 是 Agito PCSuite 运动窗口允许的最大加速度。在此窗口中，用户可以输入所需的加速度，作为该值的百分比。它保存在控制器中但不被使用，因此用户可以在忽略此值的情况下设置更高的加速度。

InTargetTol（到位容差）

项	内容
关键字（Mnemonic）	InTargetTol
CAN 编码	265
类型	读/写参数（Read / Write）
是否与轴相关	是（Yes）
数值类型	32 位整数（’int’ 32 bit）
用户单位	用户单位（In user units）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	是（Yes）
范围	0 到 2,147,483,647（32 位有符号整数最大值）
默认值	10

定义与作用

作用：

InTargetTol 表示“目标内”的公差值（误差范围），用以判断当前位置是否已达到目标位置。

在运动过程中，如果当前位置误差（目标位置与当前位置的差值）小于该公差，且持续至少 InTargetTime 毫秒，则目标状态会标记为已到达（InTargetStat 将接收到一个值，表示达到目标）。

使用场景：

运动控制中的目标达到判定条件：误差在容差范围内，并且保持稳定时间。

提高运动结束判断的精度，避免误差过大或短暂误差导致的误判。

相关参数

参数	描述
InTargetTime	误差在公差范围内持续的时间 (毫秒)
InTargetStat	判断运动目标是否达成的状态指示（已到达或未到达）

容差值：

根据实际运动精度需求设置。

数值越小，判定越严格，运动结束会更精准，但可能因微小振动或噪声出现多次未达标。

数值较大，则判定较宽松，适合误差难以完全消除的场合。

时间设置：

设定 InTargetTime，确保目标达到的稳定性。

典型值范围：10 到 100 毫秒，根据运动的速度和平稳性调整。

InTargetTime（到位持续时间）

项	内容
关键字（Mnemonic）	InTargetTime
CAN 编码	266
类型	读/写参数（Read / Write）
是否与轴相关	是（Yes）
数值类型	32 位整数（'int' 32 bit）
用户单位	毫秒（毫秒为时间单位，不需用户转换）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	是（Yes）
范围	0 至 9,999 毫秒
默认值	3

定义与作用

作用：
InTargetTime 指定目标位置被认为“已达成”的持续时间，单位为毫秒。

当当前位置误差在 InTargetTol 容差范围内时，系统会开始计时，直到误差持续在容差内的时间达到或超过 InTargetTime 毫秒，才会判定运动目标已达成（InTargetStat 触发）。

这样可以避免运动中的短暂误差或震荡导致误判，提高运动判定的可靠性。

应用场景：

精确控制系统中的目标完成判定。

在位置达到后等待一定时间确认，确保运动稳定。

时间设置：

时间越长，判定越“保守”，避免误判，但会延迟目标达成识别。

通常设置在 1 到 10 毫秒之间，根据实际控制需求调整。

PTPKeepMoving（点到点保持运动）

项	内容
关键字（Mnemonic）	PTPKeepMoving
CAN 编码	625
类型	读/写参数（Read / Write）
是否与轴相关	是（Yes）
数值类型	32 位整数（’int’ 32 bit）
用户单位	无（No user units）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
范围	0 到 1
默认值	0

定义与作用

作用：

PTPKeepMoving 控制点到点（PTP）运动中的特殊行为：

当设为 1 时，运动过程中，如果还未到达目标位置，运动会持续，不会因为目标位置的“停止”命令而中断。

当设为 0 时，到达目标位置后，运动会暂停或停止，结束运动。

LockEn（位置锁定启用）

项	内容
关键字（Mnemonic）	LockEn
CAN 编码	170
类型	读/写参数（Read / Write）
是否与轴相关	是（Yes）
数值类型	32 位整数（'int' 32 bit）
用户单位	无（No user units）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
范围	0 或 1
默认值	0

LockEn = 1 将启用位置锁定。当位置锁定启用时，指定输入的变化（顺时针运动上升，逆时针运动下降）将锁定主编码器位置。

锁定和事件是互斥功能，因此当锁定启用时，事件将自动禁用。

输入变化期间的位置值保存在“LockVal”中。位置被锁定的次数保存在“LockCntr”中。

对于增量编码器，位置由硬件锁定，因此非常准确。对于绝对编码器，在锁定输入变化后的下一个采样中读取的位置保存在“LockVal”中。

LockVal 和 LockCntr 仅在每个采样时间更新一次。如果在单个采样时间内发生多个有效输入变化，则仅记录最后一次变化。

另请参阅：

LockVal, LockCntr, LockSrc

LockCntr（位置锁定计数器）

项	内容
关键字（Mnemonic）	LockCntr
CAN 编码	172
类型	读/写参数（Read / Write）
是否与轴相关	是（Yes）
数值类型	32 位整数（'int' 32 bit）
用户单位	无（No user units）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
范围	-2,147,483,648 到 2,147,483,647
默认值	0

定义与作用

作用：

LockCntr 记录自上次启用位置锁定（LockEn 由 0 变 1 或 1 变 0 触发）以来，检测到的符合锁定源（LockSrc）条件的输入变化次数。

关键点：

通过改变 LockEn 由 0 到 1 实现锁定，或由 1 变 0 时重置计数器。

用户也可以通过直接赋值为 0 来重置这个计数器。

相关参数

参数	作用说明
LockEn	启用或禁用位置锁定
LockVal	当前锁定位置值
LockSrc	触发位置锁定的输入源
LockCntr	自上次锁定/解锁后检测到的输入变化次数

LockVal（位置锁定值）

项	内容
关键字（Mnemonic）	LockVal
CAN 编码	173
类型	只读参数（Read only）
是否与轴相关	是（Yes）
数值类型	32 位整数（’int’ 32 bit）
用户单位	用户单位（In user units）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
范围	-2,147,483,648 到 2,147,483,647
默认值	0

作用：

LockVal 保存在位置锁定期间输入变化后，最新锁定的位置信息。

对于 增量编码器：此值由硬件直接保存，极其准确。

对于 绝对编码器：在输入变化后的下一采样周期会自动保存当前位置。

应用场景：

用于记录或回溯锁定点的当前位置值，作为位置参考。

可结合 LockEn 和 LockSrc 监控输入变化对锁定位置的影响。

LockSrc（锁定功能源输入编号）

项	内容
关键字（Mnemonic）	LockSrc
CAN 编码	171
类型	读/写参数（Read / Write）
是否与轴相关	是（Yes）
数值类型	32 位整数（’int’ 32 bit）
用户单位	无（No user units）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	是（Yes）
范围	-38 到 38
默认值	1

定义与作用

作用：

LockSrc 指定用作位置锁定的输入信号编号（通常为数字编码，代表具体的数字输入）；

当此输入信号发生变化（上升或下降）时，系统会将当前位置值保存到 LockVal 和变化次数到 LockCntr。

被设置为锁定源的输入可以在 DInMode（数字输入模式）中被定义为不同的角色，比如限位开关或其他功能。

应用示例：

输入编号 3 可以既作为左限位开关，又可以作为锁定源。这一设置允许在同一物理输入上实现不同的功能，比如触发锁定或限位检测。

相关参数

参数	作用说明
LockEn	启用/禁用位置锁定
LockVal	当前锁定位置值
LockCntr	自上次启用/解除锁定以来检测到的变化次数

参数	作用说明
DInMode	数字输入的角色定义（用于分配输入信号的不同角色）

LockValTable（位置锁定值表）

项	内容
关键字（Mnemonic）	LockValTable
CAN 编码	386
类型	数组参数（Array with index range of: 1 : 65000）
访问权限	只读（Read only）
是否与轴相关	是（Yes）
数值类型	32 位整数（'int' 32 bit）
用户单位	用户单位（In user units）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
范围	-2,147,483,648 到 2,147,483,647
默认值	0

定义与作用

作用：

LockValTable 是一组保存多个锁定位置值的数组，索引范围从 1 到 65,000。

这些值可以用于存储不同状态下的锁定位置，例如不同的锁定点、位置点或历史记录。

由于是只读数组，系统自动维护，用户可以读取特定索引的锁定位置值。

应用场景：

在复杂运动控制中，用于存储多个锁定点的位置值。

实现多点锁定或备用锁定位置的管理。

通过索引方便快速访问不同的锁定状态。

LockTimeTable（位置锁定时间表）

项	内容
关键字（Mnemonic）	LockTimeTable
CAN 编码	387
类型	数组参数（Array with index range of: 1 : 65000）
访问权限	只读（Read only）
是否与轴相关	是（Yes）
数值类型	32 位整数（’int’ 32 bit）
用户单位	无（No user units）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
存储到闪存	否（No）
范围	-2,147,483,648 到 -147,483,647（注意：范围为负数，范围设置应确认）
默认值	0

定义与作用

作用：

LockTimeTable 是存储多组位置锁定时间的数组，索引从 1 到 65,000。

每个时间值定义了相应锁定点的时间（单位通常是毫秒，也可能是特定的时间单位，需根据系统定义确认）。

应用场景：

多点锁定：每个锁定点对应不同的锁定时间，有助于实现时间同步或不同锁定状态的时间管理。

UserParam（用户定义参数）

项	内容
关键字（Mnemonic）	UserParam
CAN 编码	624
类型	数组参数（Array with index range of: 1 : 250）
访问权限	读写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（’int’ 32 bit）
用户单位	无（No user units）
允许在运动中	是（Yes）
允许在电机开启时	是（Yes）
可存闪存	是（Yes）
范围	-2,147,483,648 到 2,147,483,647
默认值	0

作用

- 这是用户可以自己定义的参数数组，支持最多 250 个元素，用于存储自定义信息。
- 存放用户特定的参数、偏好或状态信息，可在运动和关机过程中保留。

UserPWM（用户 PWM 控制）

项目	内容
助记符（关键字）	UserPWM
CAN 代码	626
类型	参数（Parameter）
访问权限	读/写（Read / Write）
允许在运动中	是（Yes）
允许伺服使能	是（Yes）
索引范围（数组）	1 ： 2（支持两个用户 PWM 通道：UserPWM1 和 UserPWM2）
保存到闪存	是（Yes）
轴相关	是（Yes）
最小值	0
最大值	4095
默认值	0
用户单位	无用户单位（No user units）
实施状态	已实施（自 FW1.3.0 起支持，与 AG300 兼容）

有两个参数用于设置用户 PWM 模块，UserPWM 和 UserPWMDiv。
UserPWM 控制每个用户 PWM 输出的占空比。这是比较值，当内部计数器小于 UserPWM[N]时，输出值为 1，否则输出值为 0。这使我们能够生成一个 PWM 信号，我们可以以 1/4096（12 位）的分辨率改变占空比。
有两个 UserPWM—UserPWM1 和 UserPWM2。UserPWM1 和 UserPWM2 具有相同的 PWM 频率并完全同步。然而，它们的占空比是分别设置的。
UserPWMDiv 控制用户 PWM 模块的时钟频率。UserPWMDiv 的值在 0 到 15 之间，其中 0 使用户 PWM 时钟计数为 40MHz，1 为 20MHz，2 为 10MHz 等。它在返回 0 之前计数 4096 次（我们称之为内部计数器）。

PWM 时钟频率 = 80MHz / 2^ (UserPWMDiv + 1)

PWM 信号的频率将是：
PWM 频率 = PWM 时钟频率 / 4096
设置这两个参数后，用户可以使用离散输出选择器将 PWM 信号输出到任何输出。请参阅 DOutSelect 手册页面以获取完整说明。
注意： UserPWM 目前在 Central-I 系统中不支持，但可以根据请求轻松添加。
示例：
AUserPWM[1]=1000; 然后它将生成一个占空比为 1000/4096=24.4%的 PWM。
AUserPWMDiv= 3 ; 然后 PWM 时钟频率=80/ 2 ^ (3 + 1) MHz =5 MHz ， 结果的 PWM 频率将是：5MHz / 4096 = 1.22 KHz
使用这些设置，选择器输出将发送出 1.22 KHz 的 PWM 输出，具有 24.4%的占空比。

另请参阅：

UserPWM Div , *D O utSelect* 。

项	内容
助记符（关键字）	UserPWMDiv
CAN 代码	627
类型	参数（Parameter）
访问权限	读/写（Read / Write）
允许在运动中	是（Yes）
允许伺服使能	是（Yes）
存储到闪存	是（Yes）
轴相关	是（Yes）
最小值	0
最大值	15
默认值	9
用户单位	无用户单位（No user units）
实施状态	已实施（支持于此版本）

功能说明

示例：

AUserPWMDiv= 3 ; 然后 PWM 时钟频率=80/ 2^ (3 + 1) MHz =5 MHz ， 结果的 PWM 频率将是：5MHz / 4096 = 1.22 KHz。

另请参阅：

UserPWM , *DoutSelect* 。

应用说明

- 通过调整 **UserPWMDiv**，可以灵活调节 PWM 的频率。
- 结合 **UserPWM** 设置占空比，形成不同频率和占空比的 PWM 信号。
- PWM 信号可以通过 **DOutSelect** 选择输出到任意数字输出端（参见对应手册）。

项	内容
助记符（关键字）	MapEncoder
CAN 代码	322
类型	参数（Parameter）
数组索引范围	1 ： 3
访问权限	读 / 写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无用户单位（No user units）
是否允许运动中修改	否（No）
是否允许伺服开启时修改	否（No）
存储到闪存	是（Yes）
最小值	1
最大值	24
默认值	1

功能描述

误差映射配置：
通过设定 MapType 参数激活不同映射模式（0：禁用，1：1D，2：2D，3：3D）。
在回零前或自动相位时，MapType 必须为 0，以避免误差表的误用。
MapType 不会存到闪存中，重启后会回零，直到系统校准完成。
配置参数：
MapEncoder[1..3]：定义用于误差校正的编码器，支持：
1：主编码器
2：辅助编码器
3 及以上：其他轴的编码器（如 B 轴主/辅编码器）
MapTable[]：存储误差校正值的表格。
MapStartIndex：误差表起始索引。
MapStartPos[]：对应编码器输入的起始位置，用作误差表的参考点。
MapLength[]：每个方向的误差表点数。
MapPosGap[]：连续误差点之间的输入编码器间隔，影响插值精度。
误差映射校正原理
1D 误差映射：
读取指定编码器输入（主或辅助编码器）。
输入值小于表起点(MapStartPos[1])取第一误差值。
超出最大范围取最后误差值。

在范围内线性插值计算误差。
将误差值加到编码器的读数，实现校正。
2D 误差映射：
先对第一个编码器进行误差校正（行方向）。
再用第二个编码器输入对列方向误差进行校正。
最终通过二维线性插值计算误差校正值。
3D 误差映射：
对第一个和第二个编码器进行二维校正。
根据第三个编码器的位置，插值得到三维误差校正值（在两层之间线性插值层数）。
最后将误差应用到位置读数，实现多维精确校正。
注意事项
在自动相位（换向）及回零操作过程中，将 MapType 设置为 0（禁用误差映射），以避免误差表用错。
系统重启或重置后，MapType 会自动初始化为 0。
建议在系统完成换向和回零后，才启用误差映射（设置 MapType 为 1、2 或 3）。
示例说明：
MapEncoder[1] = 1：使用第一个编码器（主编码器）作为误差校正输入。
MapEncoder[2] = 2：第二个编码器，可能是系统的辅助编码器。
MapEncoder[3] = 3：可能是 B 轴的主编码器，用于 3D 误差映射。

MapEncoder（编码器误差映射）

项	内容
关键字（Mnemonic）	MapEncoder
CAN 代码	322
类型	参数（Parameter）
数组索引范围	1 到 3
访问权限	读写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中修改	否（No）
允许伺服开启时修改	否（No）
存储到闪存	是（Yes）
最小值	1
最大值	24
默认值	1

备注

- 只支持在静止状态下修改（不允许运动中设置）。
- 该参数支持对 1D、2D、3D 误差映射进行配置。
- 典型用途：指定使用哪个编码器进行误差校正（例如：主编码器、辅助编码器等）。

MapLength（误差映射点数）

项	内容
关键字（Mnemonic）	MapLength
CAN 代码	324
类型	参数（Parameter）
数组索引范围	1 到 3
访问权限	读写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中修改	否（No）
允许伺服开启时修改	否（No）
存储到闪存	是（Yes）
最小值	1
最大值	3,000
默认值	10

说明

- 代表每个误差映射方向上采样点的数量。
- 影响误差校正的插值精度和曲线细腻程度。
- 在配置误差映射参数时设置合适的点数，有助于获得更精准的校正结果。

MapPosGap（误差点输入编码器间隔）

项	内容
关键字（Mnemonic）	MapPosGap
CAN 代码	325
类型	参数（Parameter）
数组索引范围	1 到 3
访问权限	读写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中修改	否（No）
允许伺服开启时修改	否（No）
存储到闪存	是（Yes）
最小值	1
最大值	8,000,000
默认值	1,000

说明

- 控制连续误差点之间的输入编码器值间距。
- 影响误差映射插值的精细程度和数据点之间的距离。
- 设定合理的间隔值，有助于优化误差校正的平滑性和精度。

MapStartIndex（误差映射起始索引）

项	内容
关键字（Mnemonic）	MapStartIndex
CAN 代码	321
类型	参数（Parameter）
数组索引范围	1 到 5,000
访问权限	读写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中修改	否（No）
允许伺服开启时修改	否（No）
存储到闪存	是（Yes）
最小值	1
最大值	5,000
默认值	1

说明

- 指定误差表的起始索引，用于误差校正数据的管理和定位。
- 配合 MapTable 使用，定义误差校正数据在表中的起始位置。
- 设定合理索引，确保误差映射的正确性和一致性。

MapStartPos（误差映射起始位置）

项	内容
关键字（Mnemonic）	MapStartPos
CAN 代码	323
类型	参数（Parameter）
数组索引范围	1 到 3
访问权限	读写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中修改	否（No）
允许伺服开启时修改	否（No）
存储到闪存	是（Yes）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0

说明

- 定义误差映射起始点的编码器输入位置，作为误差表的参考基准。
- 配合 MapTable 配合使用，用于误差校正的起点位置。
- 允许正常的正负值范围，支持绝对或相对位置校正。

项	内容
关键字（Mnemonic）	MapTable
CAN 代码	326
类型	参数（Parameter）
数组索引范围	1 到 3000
访问权限	读/写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中修改	否（No）
允许伺服开启时修改	否（No）
存储到闪存	是（Yes）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0

说明

- 存储误差映射的具体校正值，最多支持 3000 个点，支持大范围误差补偿。
- 结合 MapStartIndex、MapStartPos 等参数，定义误差映射的具体数据位置。
- 支持正负值，适应各种误差校正需求。

MapErrOffset（误差映射偏移）

项	内容
关键字（Mnemonic）	MapErrOffset
CAN 代码	411
类型	参数（Parameter）
访问权限	读/写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中修改	否（No）
允许伺服开启时修改	否（No）
存储到闪存	否（No）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0

说明

- 用于误差校正的偏移补偿，可调整误差映射的整体偏差。
- 不存储到闪存，通常用于调试或临时校准。
- 支持在运行状态下调整（如果需要），但建议谨慎操作。

MapErrOffRamp（误差映射偏移斜率）

项	内容
关键字（Mnemonic）	MapErrOffRamp
CAN 代码	454
类型	参数（Parameter）
访问权限	读/写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中修改	是（Yes）
允许伺服开启时修改	是（Yes）
存储到闪存	是（Yes）
最小值	1
最大值	2,147,483,647
默认值	16,384

说明

- 作用：用于误差映射校准中为偏移提供动态调节的斜率参数，调整偏移随误差变化的速率。
- 支持运动中调整，便于现场调试。
- 设定合理值，有助于细化误差补偿的实时动态变化。

MapErrOnStep（误差映射步数）

项	内容
关键字（Mnemonic）	MapErrOnStep
CAN 代码	476
类型	参数（Parameter）
访问权限	读/写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中修改	是（Yes）
允许伺服开启时修改	是（Yes）
存储到闪存	是（Yes）
最小值	0
最大值	16,384
默认值	0

说明

- 定义误差映射开始触发的误差步数门槛，达到此值后开始校正。
- 允许在运动过程中调整，方便动态调节。
- 适合在误差逐渐累积时，控制误差映射的启动点。

GantryOn（龙门开启）

项	内容
关键字（Mnemonic）	GantryOn
CAN 代码	650
类型	参数（Parameter）
访问权限	读/写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中修改	否（No）
允许伺服开启时修改	是（Yes）
存储到闪存	否（No）
最小值	0
最大值	1
默认值	0

说明

- 0 表示“未开启”状态，1 表示“已开启”状态。
- 典型应用：控制龙门的启停状态，作为开关量控制。
- 只能在伺服驱动开启状态下修改。

GantryAccFFW（龙门加速度前反馈）

项	内容
关键字（Mnemonic）	GantryAccFFW
CAN 代码	655
类型	参数（Parameter）
访问权限	读/写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中修改	是（Yes）
允许伺服开启时修改	是（Yes）
存储到闪存	是（Yes）
最小值	0
最大值	500,000
默认值	0

说明

- 控制龙门加速度前反馈越限值，调整此参数可改善加速度控制响应。
- 支持运动中调节，适用于动态调整加速度反馈。

GantryPosGain（龙门位置增益）

项	内容
关键字（Mnemonic）	GantryPosGain
CAN 代码	654
类型	参数（Parameter）
访问权限	读/写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中修改	是（Yes）
允许伺服开启时修改	是（Yes）
存储到闪存	是（Yes）
最小值	0
最大值	100,000
默认值	100

说明

- 位置控制的增益值，调节此参数可影响龙门位置响应的敏感度和稳定性。
- 支持运动中调整，适合动态优化控制。

GantryVelGain（龙门速度增益）

项	内容
关键字（Mnemonic）	GantryVelGain
CAN 代码	656
类型	参数（Parameter）
访问权限	读/写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中修改	是（Yes）
允许伺服开启时修改	是（Yes）
存储到闪存	是（Yes）
范围	0 到 100,000
默认值	100

说明

- 调节速度控制中的增益参数，影响系统对速度输入的反应敏感度。
- 支持在运动中进行调节，适合动态优化。

GantryVelKi（龙门速度积分增益）

项	内容
关键词（Mnemonic）	GantryVelKi
CAN 代码	657
类型	参数（Parameter）
访问权限	读/写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中修改	是（Yes）
允许伺服开启时修改	是（Yes）
存储到闪存	是（Yes）
范围	0 到 100,000
默认值	100

说明

- 这是速度控制的积分增益参数，用于改善系统在速度控制中的响应和稳定性。
- 支持运动时调节，帮助动态调优。

项	内容
关键字（Mnemonic）	GantryFdbk
CAN 代码	652
类型	只读参数（Read-only）
访问权限	仅允许读取（Read only）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0
用户单位	按用户单位

简要说明：

此参数为只读，反映 MIMO 型龙门控制的反馈值。
- 计算公式：
- AGantryFdbk = (APos + BPos) / 2
- BGantryFdbk = (APos – BPos)
- 计算中包括了 GantryOffset，尽管公式未显示。
- 非“A”或“B”的参数名（以“?”开头的）无实际用途，总是返回 0。

- 该参数用于测量和监控龙门的反馈状态。
- 计算结果会在任何模式（开启或关闭）下持续更新。

GantryOffset（龙门偏移）

项	内容
关键字（Mnemonic）	GantryOffset
CAN 代码	653
类型	只读参数（Read-only）
访问权限	仅允许读取（Read only）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0
用户单位	按用户单位

总结：

- **AGantryOffset** 仅在用户将 **AGantryOn** 从 0 切换到 1 时计算一次。
- 计算公式：

AGantryOffset = APosRef - BPosRef

- 该偏移用于后续的 **GantryFdbk** 计算，以消除两个编码器初始偏差。
- 实际计算公式：
- AGantryFdbk = (APos + BPos + AGantryOffset) / 2
- BGantryFdbk = (APos - BPos - AGantryOffset)
- ?GantryOffset（以“?”开头的参数）无用，总是返回 0

- 该参数代表两个编码器之间的初始偏差，用于优化反馈的准确性。
- **只读**，在系统运行中自动更新，不可手动修改。

GantryYawRef（龙门偏航参考值）

项	内容
关键字（Mnemonic）	GantryYawRef
CAN 代码	651
类型	读写参数（Read / Write）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	-20,000
最大值	20,000
默认值	0
用户单位	按用户单位
实现状态	已实现

说明：

- 该参数用于设定龙门的偏航参考值，调节偏航角度。
- 支持在运动中调整和动态调节。

EventCntr（事件计数器）

项	内容
关键字（Mnemonic）	EventCntr
CAN 代码	186
类型	读写参数（Read / Write）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	否（No）
是否与轴相关	是（Yes）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0
用户单位	无（No user units）

说明：

- **EventCntr** 用于累计自上次 **EventOn** 以来发生的事件数量。
- **EventOn** 切换会重置计数器。
- 用户也可以手动重置此计数器。

EventSelect（事件选择）

项	内容
关键字（Mnemonic）	EventSelect
CAN 代码	317
类型	参数（Parameter）
访问权限	读/写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
范围	0 到 7
默认值	1

说明：

- 该参数用于选择特定的事件，范围从 0 到 7。
- 可以在运动中和伺服开启时调节。
- 通过改变此值，可以控制或触发不同的事件条件。

EventOn（事件开启）

项	内容
关键字（Mnemonic）	EventOn
CAN 代码	178
类型	参数（Parameter）
访问权限	读/写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	否（No）
范围	0 到 1
默认值	0

功能说明：

- 当设置 EventOn = 1 时，会加载第一个位置到位置比较器（基于 EventType，以及相关参数如 EventBegPos 和 EventEndPos），并触发事件的开始。
- **建议**：在电机位置小于第一个请求事件位置时输入 EventOn = 1，以避免异常行为。
- **互斥关系**：Event 和 Lock 功能互斥。EventOn = 1 时，系统会自动将 LockEN 置为 0。

EventNextPos（事件下一个位置）

项	内容
关键字（Mnemonic）	EventNextPos
CAN 代码	319
类型	只读参数（Read only）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	以用户单位计
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	否（No）
范围	-2,147,483,648 到 2,147,483,647
默认值	0

说明：

- 这是一个只读参数，显示当前的“下一个事件位置”值。
- 结合其他事件参数（如 EventEndPos,EventCntr）使用，用以控制事件触发和位置管理。

EventTableBeg（事件表起始位置）

项	内容
关键字（Mnemonic）	EventTableBeg
CAN 代码	184
类型	读写参数（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
范围	1 到 65,000
默认值	1

说明：

- 该参数定义事件表的起始位置。
- 支持在运动状态下调节。
- 设置将存储到设备闪存中，用于持久化配置。

EventTableEnd（事件表结束位置）

项	内容
关键字（Mnemonic）	EventTableEnd
CAN 代码	185
类型	读写参数（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
范围	1 到 65,000
默认值	1

说明：

- 定义事件表的结束位置。
- 可以在运动状态下设置，且配置会存储到闪存，确保持久化。

项	内容
关键字（Mnemonic）	EventTableSrc
CAN 代码	313
类型	读写参数（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
范围	0 到 1
默认值	0

说明：

- 控制事件表的源类型，值为 0 或 1。
- 在运动中和伺服开启时都可以调节，配置会存到闪存，适合长期设置。

EventTableSel（事件表选择）

项	内容
关键字（Mnemonic）	EventTableSel
CAN 代码	318
类型	具有索引范围的数组（Array）
索引范围	1 到 65,000
访问权限	读写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	否（No）
范围	0 到 7
默认值	1

说明

- 这是一个索引数组参数，用于选择对应的事件表。
- 允许在运动状态和伺服开启时修改，但配置不存储到闪存。

项	内容
Mnemonic （关键字）	EventTable
CAN 代码	316
类型	参数（Parameter）
索引范围	1 ： 65000
访问权限	读 / 写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（ ‘int’ ）
用户单位	以用户单位计（In user units）
运动中允许修改	是（Yes）
与电机状态关系	允许（Allowed in motion）
是否存储到闪存	否（No）
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0

定义和作用：

EventTable 为应用中用于存储一组位置点的数组，用于在运动或反馈位置达到某一位置时触发事件（如脉冲输出）。

支持的索引范围从 1 到 65000，拥有庞大的存储空间，适用于复杂路径规划和多点触发。

访问限制与存储：

- 可以在运动过程中动态读写，支持与轴运动状态共存。
- 不存储到设备闪存中，系统重启后，EventTable 需重新配置。

项	内容
Mnemonic （关键字）	EventPulseWid
CAN 代码	179
类型	参数（Parameter）
访问权限	读 / 写（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（ ‘int’ ）
用户单位	无（No user units）
运动中允许修改	是（Yes）
与电机状态关系	允许（允许运动中修改，允许伺服开启时修改）
是否存储到闪存	是（Yes）
范围	-10,000,000 到 10,000,000
默认值	50

参数说明：

定义和作用：

EventPulseWid 用于设置事件信号脉冲的宽度，即触发事件时输出脉冲信号的持续时间（单位为整数，无用户单位）。

使用场景：

调节脉冲宽度以达到不同的信号持续时间，适用于同步控制、信号强弱调节等场合。

项	内容
Mnemonic （关键字）	EventType
CAN 代码	180
类型	参数（Parameter）
访问权限	读 / 写（Read / Write）
允许在运动中修改	是（Yes）
允许伺服开启时修改	是（Yes）
存储到闪存	是（Yes）
是否与轴相关	是（Yes）
范围	0 到 2
默认值	0
用户单位	无（No user units）

事件是在指定输出上生成的脉冲，当实际反馈位置等于所需比较位置时生成。 EventType 确定不同的比较选项：

- EventType = 0: 当反馈位置等于 EventBegPos 中的位置时，将生成一个脉冲。
- EventType = 1: 按间隔事件。第一个脉冲在位置等于 EventBegPos 时生成。每次通过 EventGap 定义的距离时生成另一个脉冲。当位置超过 EventEndPos 时，脉冲停止生成。
- EventType = 2: 按表事件。将应生成事件的位置表输入到 GenData [] 数组中。 ETStart 是事件表开始的索引。 ETEnd 是事件表结束的索引。表中的位置必须从低到高排序。

相关参数：

- EventOn：控制事件的开启与关闭。
- EventGap：定义在 EventType=1 时，两个事件之间的距离。
- EventEndPos：定义事件终止触发的位置。
- EventTableBeg 和 EventTableEnd：定义表格位置的起止索引。

EventBegPos（事件开始位置）

项	内容
Mnemonic （关键字）	EventBegPos
CAN 代码	181
类型	参数（Parameter）
访问权限	读 / 写（Read / Write）
允许在运动中修改	是（Yes）
允许伺服开启时修改	是（Yes）
存储到闪存	是（Yes）
是否与轴相关	是（Yes）
范围	-2,147,483,648 到 2,147,483,647
默认值	0
用户单位	以用户单位计

参数说明：

定义与用途：

- EventBegPos 指定第一个事件产生的反馈位置，适用于**事件类型 1（通过间隔）和类型 2（通过位置表）**。
- 在事件类型 1 中，第一次事件在反馈位置达到 EventBegPos 时产生。
- 在事件类型 2 中，定义了事件开始的基础位置，作为表中第一个触发点的参考值。

实际应用：

- 配合 EventType 参数，实现精准的事件触发控制。

建议：

- 设置合理的起始位置，避免事件触发不到或过早触发。
- 位置值可以为负或正，根据实际运动目标进行配置。

项	内容
Mnemonic （关键字）	EventEndPos
CAN 代码	183
类型	参数（Parameter）
访问权限	读 / 写（Read / Write）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
是否与轴相关	是（Yes）
范围	-2,147,483,648 到 2,147,483,647
默认值	0
用户单位	在用户单位（In user units）

定义和用途：

- EventEndPos 定义事件产生的最大反馈位置。在使用 EventType=2（由表驱动）时，此位置之后不会再产生事件。
- 例如：当事件类型设置为 2，EventBegPos 设为 1000，EventGap 为 2000，EventEndPos 设为 8000，系统会在位置达到 1000、3000、5000 和 7000 时产生事件（依赖于 EventPulseWid 设置的脉冲宽度），在过了 8000 位置后，不会再产生事件。

使用建议：

- 在运动前，将 EventOn 设置为小于 EventBegPos 的位置，确保事件开始前系统不误动作。
- 在达到 EventEndPos 后，建议将 EventOn 关闭或重新设置，以便重新开始事件生成。

示例说明：

- 假设：
  - EventType=2
  - EventBegPos=1000
  - EventGap=2000
  - EventEndPos=8000
  - EventOn=1 在位置小于 1000 的点设置

- 系统会在位置：1000、3000、5000、7000 出现事件，直到超出 8000，此后不会再次产生事件，直到手动重新开启。

项	内容
Mnemonic （关键字）	EventGap
CAN 代码	182
类型	参数（Parameter）
访问权限	读 / 写（Read / Write）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
是否与轴相关	是（Yes）
范围	-2,147,483,648 到 2,147,483,647
默认值	0
用户单位	在用户单位（In user units）

参数说明：

定义和作用：

- EventGap 表示两次事件产生之间的距离（或位置差）值。
- 在 EventType=1（通过间隔）或 EventType=2（通过位置表）时有效。
- 设置较小的 EventGap 数值会导致事件频繁发生，尤其在高速运动时，可能会与 EventPulseWid 设置的脉冲宽度重叠，影响信号的正确性。

使用建议：

- 选择合适的 EventGap，避免过于频繁的触发或脉冲叠加。
- 在高速运动或长脉冲宽度设置时，考虑增大 EventGap 以确保信号清晰。

存储与调节：

- 支持在运动中动态调整，配置会存入闪存。
- 该参数的调整能影响事件触发的频率和响应。

示例：

- 如果设定：
  - EventType=1
  - EventBegPos=1000
  - EventGap=2000
  - EventEndPos=8000
  - EventOn=1

- 运动过程中，系统将在位置：1000、3000、5000、7000 产生事件，脉冲持续时间由 EventPulseWid 配置决定。

项目	内容
关键字（Mnemonic）	EventTableCor
CAN 代码	315
类型	只读参数（Read only）
索引范围	1 到 65000
是否与轴相关	是
值类型	32 位整型（int）
用户单位	以用户单位为准
运动中允许读取	是
是否存储到闪存	否
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0

说明

作用：
这个参数用于提供关于事件表位置的偏移校正，帮助进行细微调节，以确保事件触发位置的精度。

使用场景：
常用于调试或校准运动中的事件触发点，或者动态调整触发偏差。

注意事项：
由于是只读参数，不能在运动中修改，只能读取，用于监控或调试。

OperationMode（操作模式）

项目	内容
关键字（Mnemonic）	OperationMode
CAN 代码	78
类型	参数（Parameter）
访问权限	读 / 写（Read / Write）
允许在运动中	不允许（No）
允许伺服开启时	不允许（No）
存储到闪存	是（Yes）
是否与轴相关	是（Yes）
范围	1 到 4
默认值	3
用户单位	无（No user units）

操作模式详解

值	操作模式	说明
1	仅电流控制	仅控制电流，电流参考值通过模拟输入设定。
2	速度控制	控制速度和电流，速度参考值通过模拟输入输入。
3	位置控制（默认）	这是默认的运动模式。在此模式下，所有控制回路都处于活动状态。位置轮廓器根据用户设定的运动要求生成位置命令。用户可以使用不同的运动模式来选择输入位置目标命令或速度命令。
4	保留或备用	（没有详细说明，常设置为默认 3）

配置建议：
默认操作模式为位置控制（值 3），适合大多数运动应用。
如果只需电流控制或速度控制，可根据需要切换对应值（1 或 2）。

ModeSwitchPos（模式切换位置）

项目	内容
关键字（Mnemonic）	ModeSwitchPos
CAN 代码	438
类型	只读参数（Read only）
索引范围	1 到 2
是否与轴相关	是
值类型	32 位整型（int）
用户单位	在用户单位（In user units）
运动中允许读取	是
是否存储到闪存	否
最小值	-2,147,483,648
最大值	2,147,483,648
默认值	0

上次切换到力(电流)操作模式的位置。 值为 0 可能意味着自开机以来没有发生切换

PosPosFlag（位置对标志）

项目	内容
关键字（Mnemonic）	PosPosFlag
CAN 代码	328
类型	读写参数（Read / Write）
是否与轴相关	是
值类型	32 位整型（int）
用户单位	无（No user units）
运动中允许	是
伺服开启时允许	是
存储到闪存	是
范围	0 到 2
默认值	0

说明

力 模式 -> 位置模式

;定义在通过位置阈值时自动切换到位置操作模式

0-禁用

1-在位置 <阈值时

2-在位置>阈值时

PosPosTh（位置对阈值）

项目	内容
关键字（Mnemonic）	PosPosTh
CAN 代码	329
类型	读写参数（Read / Write）
允许在运动中	是
允许伺服开启时	是
存储到闪存	是
是否与轴相关	是
范围	-2,147,483,648 到 2,147,483,647
默认值	0
用户单位	在用户单位（In user units）

当 **PosPosFlag 不为 0 是， PosPosTh** 达到自动切换到位置模式的位置

CurrGain（电流增益）

项目	内容
关键字（Mnemonic）	CurrGain
CAN 代码	104
类型	读写参数（Read / Write）
允许在运动中	是
允许伺服开启时	是
存储到闪存	是
是否与轴相关	是
范围	0 到 200,000
默认值	0
用户单位	无（No user units）

Agito 控制器实现了用于电流闭环控制的 PI 控制滤波器。

实现了两个控制回路。一个用于电机 A 相电流（ Ia ）和一个用于电机 B 相电流（ Ib ）。

CurrGain 是此 PI 的增益。

相关方程为：

CurrRef = … 速度控制滤波器的输出。请参阅 VelGain 和 VelKi 的文档。

IaRef = commutation(CurrRef)

IaErr = IaRef – Ia

IaIntegral = IaIntegral + IaErr * CurrKi * 0.001

IaIntegral 然后被参数 MaxPWM 饱和。

Temp = IaIntegral + IaErr

Va = Temp * CurrGain * 0.001

Va 然后被参数 MaxPWM 饱和。

（类似的控制回路也对电机 B 相电流执行，当然，换向适当偏移）。

CurrRef, IaRef, Ia, IaErr 均以 [mA] 为单位。

Va 以内部单位表示，具有代表 100% PWM 占空比的产品相关值。如果需要您使用的产品的确切内部值，请咨询 Agito。

另请参阅：

CurrKi, Ia, Ib, CurrRef, IaRef, IbRef, IaErr, IbErr, Va, Vb, Vc, MaxPWM 和 ControlMode。

Agito Product1 用户注意：

在此产品中，电流控制算法使用略有不同的方程实现（思想相同，但有不同的缩放和略有不同的方程组织）。如果您需要此情况的确切方程，请联系 Agito。

CurrKi（电流积分）

项目	内容
关键字（Mnemonic）	CurrKi
CAN 代码	105
类型	读写参数（Read / Write）
允许在运动中	是
允许伺服开启时	是
存储到闪存	是
是否与轴相关	是
范围	0 至 200,000
默认值	0
用户单位	无（No user units）

Agito 控制器实现了用于电流闭环控制的 PI 控制滤波器。

实现了两个控制回路。一个用于电机 A 相电流（ Ia ）和一个用于电机 B 相电流（ Ib ）。

CurrKi 是此 PI 的积分项增益。

相关方程为：

CurrRef = … 速度控制滤波器的输出。请参阅 VelGain 和 VelKi 的文档。

IaRef = commutation(CurrRef)

IaErr = IaRef – Ia

IaIntegral = IaIntegral + IaErr * CurrKi * 0.001

IaIntegral 然后被参数 MaxPWM 饱和。

Temp = IaIntegral + IaErr

Va = Temp * CurrGain * 0.001

Va 然后被参数 MaxPWM 饱和。

（类似的控制回路也对电机 B 相电流执行，当然，换向适当偏移）。

CurrRef, IaRef, Ia, IaErr 均以 [mA] 为单位。

Va 以内部单位表示，具有代表 100% PWM 占空比的产品相关值。如果需要您使用的产品的确切内部值，请咨询 Agito。

另请参阅：

CurrGain, Ia, Ib, CurrRef, IaRef, IbRef, IaErr, IbErr, Va, Vb, Vc, MaxPWM 和 ControlMode。

Agito Product1 用户注意：

在此产品中，电流控制算法使用略有不同的方程实现（思想相同，但有不同的缩放和略有不同的方程组织）。如果您需要此情况的确切方程，请联系 Agito。

项目	内容
关键字（Mnemonic）	MotorCurr
CAN 代码	8
类型	只读参数（Read only）
允许在运动中	是
允许伺服开启时	是
存储到闪存	否
是否与轴相关	是
范围	-2,147,483,648 到 2,147,483,647
默认值	0
用户单位	无（No user units）

功能：

- MotorCurr 表示电机的总电流，包括所有相的电流总和，单位为毫安（mA）。
- 主要用于监控电机的工作状态，检测过载或异常电流。

CurrPosErrTh（位置误差阈值）

项目	内容
关键字（Mnemonic）	CurrPosErrTh
CAN 代码	337
类型	读写参数（Read / Write）
允许在运动中	是
允许伺服开启时	是
存储到闪存	否
是否与轴相关	是
范围	-327,680 到 327,680
默认值	0
用户单位	在用户单位（In user units）

必须满足的全局阈值,以便切换到电流或力操作模式。 如果位置参考(PosRef)大于或小于阈值,则可以进行切换(如果触发了其他任何阈值) 。

需要参考 CurrPosThDir; 'int' 32 bit;定义阈值方向（大于或小于)。值为 0 绕过此条件

-1-在 PosRef < 阈值时

0 - 绕过条件

1 - 在 PosRef > 阈值时

CurrCurrTh（电流阈值）

项目	内容
关键字（Mnemonic）	CurrCurrTh
CAN 代码	339
类型	读写参数（Read / Write）
允许在运动中	是
允许伺服开启时	是
存储到闪存	否
是否与轴相关	是
范围	-64,000 到 64,000
默认值	0
用户单位	无（No user units）

开环力控切换力：如果电流参考(CurrRef)大于或小于阈值,则用于切换到电流操作模式的阈值。

CurrCurrThDir（开环力控电流阈值方向）

项目	内容
关键字（Mnemonic）	CurrCurrThDir
CAN 代码	343
类型	读写参数（Read / Write）
允许在运动中	是
允许伺服开启时	是
存储到闪存	是
是否与轴相关	是
范围	0 到 1
默认值	0
用户单位	无（No user units）

开环力控：CurrCurrThDir; 'int' 32 bit;定义阈值方向（大于或小于）。值为 0 绕过此条件

0-在 CurrRef > 阈值时

1-在 CurrRef < 阈值时

CurrAInTh（开环力控力反馈阈值）

项目	内容
关键字（Mnemonic）	CurrAInTh
CAN 代码	338
类型	读写参数（Read / Write）
允许在运动中	是
允许伺服开启时	是
存储到闪存	否
是否与轴相关	是
范围	-100,000 至 100,000
默认值	0
用户单位	无（No user units）

开环力控：达到自动切换到力模式的力反馈值 0 值以避免这种类型的切换。正值或负值定义阈值方向

项目	内容
关键字（Mnemonic）	CurrCmdSrc
CAN 代码	330
类型	读写参数（Read / Write）
允许在运动中	是
允许伺服开启时	是
存储到闪存	是
是否与轴相关	是
范围	0 到 2
默认值	0
用户单位	无（No user units）

开环力控电流命令源

作用：

- 控制当前命令的来源，可用值包括：

0: 模拟输入（AInMode 定义的输入）

1: 用户自定义数值或时间

2: 用户定义值/时间（插值，未来支持，目前与 1 相同）

应用说明：

- 通过设置不同的值，可以动态切换电流命令的来源，实现多种控制策略。
- 在实际操作中，通常设为 0（模拟输入）或 1（用户定义）。

项目	内容
关键字（Mnemonic）	CurrCmdSlope
CAN 代码	568
类型	数组参数（Index 范围 1..20）
访问权限	读写
是否与轴相关	是
值类型	32 位整数（’int’ 32 bit）
允许在运动中	是
允许伺服开启时	是
存储到闪存	是
范围	1 到 2,147,483,647
默认值	100
用户单位	无（No user units）

开环力控：

电流 命令源： 1-分段的 电流 命令值

作用：

- 定义每个控制周期内，电流命令变化的最大坡度（即变化速率限制）。
- 通过限制电流变化率，有助于减少电流突变，保护硬件和改善运动平稳性。

应用场景：

- 控制电流逐步变化，从而控制加减速的平滑程度。
- 在多轴同步运动，避免电流突变导致运动不稳定。

特点：

- 数组长度为 20，意味着可以为最多 20 个控制点或多段定义不同的坡度，增强灵活性。

项目	内容
关键字（Mnemonic）	CurrCmdHTime
CAN 代码	332
类型	数组参数（索引范围 1..20）
访问权限	读写
是否与轴相关	是
值类型	32 位整数（’int’ 32 bit）
允许在运动中	是
允许伺服开启时	是
存储到闪存	是
范围	-122,070,306 到 122,070,306
默认值	0
用户单位	无（未定义）

开环力控：

电流 命令源： 1-分段的 电流 命令值

作用：

- 设定电流命令在保持状态（保持时间）内持续保持的时间（单位未明确，通常为毫秒或控制周期）。
- 控制电流持续时间，影响运动的平滑性和响应性。

应用场景：

- 调整电流命令的持续时间，从而优化控制策略。
- 配合坡度参数，实现平滑、渐变的电流变化。

项目	内容
关键字（Mnemonic）	CurrCmdVal
CAN 代码	331
类型	数组参数（索引范围 1..20）
访问权限	读写
允许在运动中	是
允许伺服开启时	是
存储到闪存	是
是否与轴相关	是
范围	根据远程单元（动态，依赖实际硬件和配置）
默认值	根据远程单元（动态）
用户单位	无（No user units）
实现状态	已实现

开环力控：

电流 命令源： 1-分段的 电流 命令值

作用：

- 设置对应索引（1 到 20）位置上的电流命令值，控制实际施加的电流。
- 非常关键的参数，用于运动控制中的具体电流指令。

应用场景：

- 在运动控制中动态设置多段电流值，用于实现精确的电流调节和运动优化。
- 配合控制状态机或运动模式切换，实现复杂的运动逻辑。

SpeedChgOn（力控慢速接近开关）

项目	内容
关键字（Mnemonic）	SpeedChgOn
CAN 代码	345
类型	读写参数（Read / Write）
允许在运动中	是
允许伺服开启时	是
存储到闪存	否
是否与轴相关	是
范围	0（禁用）到 1（启用）
默认值	0
用户单位	无（No user units）

力控默认慢速接近开关

作用：

- 力控是否在特定位点（Position）自动切换到预设的新速度（Speed）。
- 当设置为 1（启用）时，到达设定位置时会自动改变速度。
- 关闭（0）则不进行自动速度切换。

参考参数：

- SpeedChgNew：定义新速度。
- SpeedChgPos：定义触发位置。
- SpeedChgDir：定义切换方向。

SpeedChgPos（力控慢速接近位置）

项	内容
关键字（Mnemonic）	SpeedChgPos
CAN 代码	346
类型	读写参数（Read / Write）
是否与轴相关	是（Yes）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
用户单位	以用户单位（In user units）
范围	-2,147,483,648 到 2,147,483,647
默认值	1,000
实现状态	已实现

功能：

力控默认慢速接近位置

- **作用**：设置在特定位置触发速度变换，变到 SpeedChgNew 指定的速度。
- **用法**：在运动路径上指定某一位置时，速度会变更为预设的新速度。
- 结合 SpeedChgNew（变速值）使用。

SpeedChgDir（力控慢速接近方向）

项	内容
关键字（Mnemonic）	SpeedChgDir
CAN 代码	347
类型	读写参数（Read / Write）
是否与轴相关	是（Yes）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
用户单位	无（No user units）
范围	0 或 1
默认值	0
实现状态	已实现

功能：

力控默认慢速接近方向

- **作用**：决定触发速度变更的方向。

0：当当前位置大于设定位置（SpeedChgPos）时，将速度变更为 SpeedChgNew。

1：当当前位置小于设定位置时，变更为新速度。

- 结合 SpeedChgPos 和 SpeedChgNew 使用，实现自动速度切换。

SpeedChgNew（力控慢速接近速度值）

项	内容
关键字（Mnemonic）	SpeedChgNew
CAN 代码	344
类型	读写参数（Read / Write）
是否与轴相关	是（Yes）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
用户单位	以用户单位（In user units）
范围	-60,000,000 到 60,000,000
默认值	10,000
实现状态	已实现

功能

力控默认慢速接近速度值

- **作用**：设置变速操作中的新速度值。
- 结合 SpeedChgPos,SpeedChgDir，实现自动或手动切换速度。

ForceRef（力控参考值）

项	内容
关键字（Mnemonic）	ForceRef
CAN 代码	581
类型	只读参数（Read-only）
是否与轴相关	是（Yes）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	否（No）
用户单位	无（No user units）
范围	-2,147,483,648 到 2,147,483,647
默认值	0
实现状态	已实现

功能：

- **作用**：显示或读取当前的强制参考值，通常用于调试或监控。
- **注意**：此参数为只读，不能直接修改。

Force（力值）

项	内容
关键字（Mnemonic）	Force
CAN 代码	582
类型	只读参数（Read-only）
是否与轴相关	是（Yes）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	否（No）
用户单位	无（No user units）
范围	-2,147,483,648 到 2,147,483,647
默认值	0
实现状态	已实现

功能说明：

- **作用**：显示或监控当前的力值,用于力控制调节，不能直接修改。
- **用途**：结合力控制算法实现对负载的精确调节，作为内部状态参数。

项	内容
关键字（Mnemonic）	ForceErr
CAN 代码	583
类型	只读参数（Read-only）
是否与轴相关	是（Yes）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	否（No）
用户单位	无（No user units）
范围	-2,147,483,648 到 2,147,483,647
默认值	0
实现状态	已实现

功能说明：

- **作用**：显示当前的力误差，用于监控和调节力控制精度。
- **注意**：这是一个监测参数，不能手动修改。

ForceGain（力增益）

项	内容
关键字（Mnemonic）	ForceGain
CAN 代码	577
类型	读写参数（Read / Write）
是否与轴相关	是（Yes）
值类型	32 位整数（int）
用户单位	无（No user units）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
范围	0 到 2,147,483,647
默认值	0

功能说明：

- **作用**：调整力反馈控制中的增益参数，数值越大，力控制响应越敏感。
- **应用**：结合力传感器和力控制算法，优化调节效果。

项	内容
关键字（Mnemonic）	ForceKi
CAN 代码	578
类型	读写参数（Read / Write）
是否与轴相关	是（Yes）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
范围	0 到 2,147,483,647
默认值	0

功能说明

- **作用**：调节力控制中的积分系数，用于减缓误差的累积，提高系统稳定性。
- **应用**：结合力误差（ForceErr）进行闭环控制，提高调节效果。

项	内容
关键字（Mnemonic）	ForceKd
CAN 代码	588
类型	读写参数（Read / Write）
是否与轴相关	是（Yes）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
范围	0 到 1,000,000
默认值	0

功能说明：

- **作用**：调节力控制的微分系数，用于抑制振荡，提高响应稳定性。
- **应用**：微分项帮助系统减缓突变或振荡，提高力控制的平稳性。

项	内容
关键字（Mnemonic）	ForceFFW
CAN 代码	589
类型	32 位整数（int）
是否与轴相关	是（Yes）
用户单位	无（No user units）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
范围	0 到 1,000,000
默认值	0

功能说明：

- **作用**：调节力前馈补偿系数，用于提前补偿预期的负载或阻抗，提升控制响应速度和精度。
- **应用**：在需要快速响应和精确力控制的场景中调整此参数。

ForceVelFFW（力速度前馈）

项	内容
关键字（Mnemonic）	ForceVelFFW
CAN 代码	580
类型	32 位整数（int）
是否与轴相关	是（Yes）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
范围	-2,147,483,648 到 2,147,483,647
默认值	0

功能说明：

- **作用**：设置速度前馈补偿，提高力控制在变速状态下的响应速度和稳定性。
- **应用**：适用于需要快速响应负载变化的动态场景。

项	内容
关键字（Mnemonic）	ForceRefFilt
CAN 代码	586
类型	读写参数（Read / Write）
是否与轴相关	是（Yes）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
范围	1 到 500,000
默认值	10,000

功能说明：

对力命令应用的一级低通滤波器，

- **作用**：滤除高频噪声，平滑力反馈参考信号，确保控制稳定。
- **应用**：调节滤波强度，平衡响应速度与噪声抑制。

MaxForceErr（闭环力控最大力误差）

项	内容
关键字（Mnemonic）	MaxForceErr
CAN 代码	585
类型	读写参数（Read / Write）
是否与轴相关	是（Yes）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
范围	0 到 32,768
默认值	2,000

功能说明：

- **作用**：设置最大允许的力误差值，以防过大偏差引起异常或保护系统。
- **应用**：用于限制力偏差范围，提高系统安全性和稳定性。

MaxForceErrOL（最大力误差开环）

项	内容
关键字（Mnemonic）	MaxForceErrOL
CAN 代码	591
类型	32 位整数（int）
是否与轴相关	是（Yes）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
范围	0 到 327,680
默认值	50,000

功能说明：

- **作用**：限制开环状态下的最大力误差，以确保安全和系统稳定。
- **应用**：特别在开环控制或无反馈情况下设置误差阈值。

ForcePosErrTh（闭环力控力位置误差阈值）

项	内容
关键字（Mnemonic）	ForcePosErrTh
CAN 代码	576
类型	读写参数（Read / Write）
是否与轴相关	是（Yes）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	否（No）
范围	-327,680 到 327,680
默认值	0
用户单位	采用用户设定单位（In user units）

功能说明：

达到自动切换到力模式的位置误差。当和 ForceAInTh 满足其中一个就切换到力控

0 值避色这种类的切拖

正值或负值定义阈值的方向

ForceAInTh（闭环力控力反馈阈值）

项	内容
关键字（Mnemonic）	ForceAInTh
CAN 代码	584
类型	读写参数（Read / Write）
是否与轴相关	是（Yes）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	否（No）
范围	-100,000 到 100,000
默认值	0
用户单位	无（No user units）

功能说明：

闭环力控

达到自动切换到力模式的力误差。当和 ForcePosErrTh 满足其中一个就切换到力控

0 值避色这种类的切拖

正值或负值定义阈值的方向

ForceCmdSrc（闭环力控力命令源）

项	内容
关键字（Mnemonic）	ForceCmdSrc
CAN 代码	570
类型	读写参数（Read / Write）
是否与轴相关	是（Yes）
允许在运动中	是（Yes）
允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
范围	0 到 2
默认值	0
用户单位	无（No user units）

ForceCmdSrc（力命令源）数值定义：

- 0：模拟输入（Analog Input）
- 1：调度力命令（Scheduled Force Command）
- 2：插值调度力命令（Interpolated Scheduled Force Command）

ForceCmdSlope（闭环力命令斜率）

项	内容
关键字（Mnemonic）	ForceCmdSlope
CAN 代码	569
类型	32 位整数（int）
数组索引范围	1 到 20
是否与轴相关	是（Yes）
是否允许在运动中	是（Yes）
是否允许伺服开启时	是（Yes）
存储到闪存	是（Yes）
范围	1 到 2,147,483,647
默认值	100

功能说明：

- **作用**：限制力指令变化的最大速率（斜率），防止突变引起系统不稳。
- **应用**：调节以确保力控制平滑过渡。

ForceCmdHTime（闭环力命令保持时间）

项目	内容
关键字（Mnemonic）	ForceCmdHTime
CAN 代码	572
类型	数组参数（索引范围 1..20）
访问权限	读写
是否与轴相关	是
允许在运动中	是
允许伺服开启时	是
存储到闪存	是
范围	-122,070,306 到 122,070,306
默认值	0
用户单位	无（No user units）
实现状态	已实现

功能简介

作用：

- 设置力命令（Force Command）的保持时间，即 Command Change 发生后，强制命令持续作用的时间（单位未注明，通常为控制周期或毫秒）。
- 多个段可定义不同时间，增强控制灵活性。

项目	内容
关键字（Mnemonic）	ForceCmdHTime
CAN 代码	572
类型	数组参数（索引范围 1..20）
访问权限	读写
是否与轴相关	是
允许在运动中	是
允许伺服开启时	是
存储到闪存	是
范围	-122,070,306 到 122,070,306
默认值	0
用户单位	无（No user units）
实现状态	已实现

功能简介

作用：

- 设置力命令保持的时间，确保强制操作持续一定时间以生效。
- 数组支持多段时间定义，增强控制灵活性。

项目	内容
关键字（Mnemonic）	ForceCmdVal
CAN 代码	571
类型	数组参数（索引范围 1..20）
访问权限	读写
允许在运动中	是
允许伺服开启时	是
存储到闪存	是
是否与轴相关	是
范围	-16,000 到 16,000
默认值	0
用户单位	无（No user units）
实现状态	已实现

作用：

- 设置在非模拟输入（ForceCmdSrc 非 0）情况下的力命令值。
- 数组格式支持在不同阶段或不同条件下设置多个值（最多 20 段），便于实现分阶段控制。

说明：

- 该参数配合 ForceCmdSrc 控制方式，用于施加实际的力（如电流或扭矩等，具体取决于硬件实现）。

注意事项：

- 力值范围为-16000 到 16000，可以调节力的大小和方向。

项目	内容
关键字（Mnemonic）	HomingOn
CAN 代码	340
类型	32 位整数（int）
访问权限	读/写
是否与轴相关	是
允许在运动中	否
允许伺服开启时	是
存储到闪存	否
范围	0（未启动回零）或 1（开始回零）
默认值	0

功能简述

作用：

- 控制是否启动回零（Homing）过程。
- 用户在需要进行回零操作时，将此参数设置为 1，控制器会根据 HomingDef[]定义启动对应的回零策略。

注意事项：

- 仅在设置为 1 时，控制器开始执行回零流程。
- 回零完成或中止后，系统会自动将此参数清零。
- 这个参数不能在运动中设置（不能在运动过程中启动回零）。

Agito 的控制器支持内置的回零过程，用户可以“编程”以创建最适合应用的回零过程。

回零过程由两个参数控制：HomingOn 参数和 HomingDef[]数组参数。

回零过程的状态在 HomingStat 参数中报告。

HomingOn 参数在上电或重置时被清除为“0”。一旦用户将其设置为“1”，控制器将根据 HomingDef 数组参数中的定义开始回零过程，并在过程中报告其状态。

回零过程完成后（无论成功与否），控制器将清除 HomingOn 参数。

回零过程由多个步骤组成。每个步骤的数量和要执行的操作（移动到限制、移动到索引、设置位置……）由 HomingDef[]数组参数中的值定义，详细描述如下。

HomingStep 关键字可用于检查当前活动的回零过程步骤。在上电后未尝试执行回零之前为 0。回零完成后（无论成功与否），它将等于最后执行的步骤（可用于分析未能完成回零过程的原因）。

在最后一步成功完成后，回零完成，控制器将清除 HomingOn 参数。

在每个步骤中，可能会发生一些错误（取决于步骤类型/操作）。如果发生错误，回零过程将中止，HomingOn 将被清除，HomingStat 将报告合适的值，如下所列。

回零过程步骤的最大数量是 HomingDef[]数组的大小除以 10。通常在 Agito 的控制器中，HomingDef[]的大小为 150，这意味着用户可以创建最多包含 15 个步骤的回零过程。

HomingStat 可能报告以下状态之一：

HomingStat 值	含义
0	上电或重置后未进行回零
正值（不为 100）	回零正在进行中。HomingStat 值反映了回零过程中当前处理的步骤编号。 另请参阅上面的 HomingStep。
-1	由于 HomingDef[]数组参数中的参数错误，回零过程失败（并中止）（在回零过程中每个步骤开始时检查与每个步骤相关的参数）。
-2	由于某个回零步骤的超时，回零过程失败（并中止）。每个步骤（如果相关）都定义了一个超时。如果此超时过去且步骤尚未完成，则为错误。
-3	由于意外的电机关闭，回零过程失败（并中止）。这意味着在某个回零步骤中，由于某些故障（在 ConFlt 的值中反映），轴被禁用，步骤无法完成。
-4	由于错误的运动原因，回零过程失败（并中止）。这意味着在回零过程中的某个步骤，期望的运动结束原因（RLS、索引、达到目标……）遇到了不同的运动结束原因。
-5	由于错误的步骤类型，回零过程失败（并中止）。这意味着回零过程到达了一个步骤，其类型（在 HomingDef[]数组参数中定义）无法识别。
-6	由于运动中，回零过程失败（并中止）。这意味着回零过程检测到轴在运动中，而它正在启动一个新步骤。
-7	由于步骤过多，回零过程失败（并中止）。如果回零过程到达 HomingDef[]数组参数中定义的最后一步，并且这不是回零结束步骤，则会发生此错误。
-8	由于意外的限制，回零过程失败（并中止）。此错误仅与某个回零步骤类型相关：“检查是否超出限制”。请参阅下面的更多详细信息。
-9	由于无法执行 SetPosition（错误映射激活？自动增益激活？值超出软件位置限制？），回零过程失败（并中止）。此错误仅与某个回零步骤类型相关：“设置位置”。请参阅下面的更多详细信息。
100	回零过程已成功完成。

如上所述，HomingDef[]数组参数用于定义回零过程，通过定义回零步骤以及每个步骤的类型和参数。

HomingDef[1]定义了第一步的类型。

HomingDef[2-10]定义了此步骤的参数。

实际使用的参数数量及每个参数的含义取决于步骤类型。

同样，HomingDef[11-20]定义了第二步。HomingDef[21-30]定义了第二步。

在文本文件中定义回零过程参数非常方便（*.par，参数文件），可以下载（使用 PCSuite）到控制器以定义回零过程。以下是此类文件的示例（给定应用的回零过程定义），后面将详细描述支持的步骤类型及每种类型的参数：

```
//
// 1. Go fast into reverse limit
//
AHomingDef[1]=1
AHomingDef[2]=-10000
AHomingDef[3]=10000000
AHomingDef[4]=10000000
AHomingDef[5]=163840
//
// 2. Wait 100ms
//
AHomingDef[11]=7
AHomingDef[12]=1638
//
// 3. Jog forward slow to index
//
AHomingDef[21]=4
AHomingDef[22]=1000
AHomingDef[23]=1000000
AHomingDef[24]=1000000
AHomingDef[25]=163840
//
// 4. Wait 100ms
//
AHomingDef[31]=7
AHomingDef[32]=1638
//
// 5. Set position to 0
//
AHomingDef[41]=6
AHomingDef[42]=0
//
// 6. End of homing
//
AHomingDef[51]=0
```

用户还可以使用 PCSuite 的回零工具（窗口），该工具提供了回零过程和参数的简便配置，以及内置的回零场景。

现在需要定义 HomingDef[]内容的详细信息，该内容定义了回零过程：

下面的描述将涉及 HomingDef[1-10]。然而，同样适用于 HomingDef[11-20]（第二步）等等。

HomingDef[1]定义了步骤的类型。下表显示了可用的类型：

HomingDef[1]（或[11]，[21] …）值	步骤类型
0	结束回零。这必须是最后一步。
1	JOG（jogging）进入限制。
2	检查是否超出限制
3	相对 PTP（PTP）移动
4	JOG 到索引
5	移动到索引位置
6	设置位置 （ *** ）
7	等待 N 个样本
8	启用（或禁用）电机
9	移动到硬停止（由电机堵转检测） （ *** ）
10	移动到硬停止（由高位置误差检测） （ *** ）
11	JOG 到 Home 离散输入的变化
12	绝对 PTP（PTP）移动
13	设置位置软件限制（RevPLim 和 FwdPLim 参数）
14	配置位置锁定
15	JOG 到锁定（通过硬件索引）
16	移动到锁定位置

（ *** ）这些回零步骤类型在位置值设置方面有一些限制。请仔细阅读下面的描述。

下表介绍了每种类型及其参数：

步骤类型	步骤描述	HomingDef[2-10]的步骤参数（或[12-20]，[22-30] …）
结束回零。这必须是最后一步。	回零过程将停止。到达此步骤意味着回零已成功完成。	无参数

步骤类型	步骤描述	HomingDef[2-10]的步骤参数 （或[12-20]，[22-30] …）
JOG（JOG）进入限制。	以步骤参数中定义的速度和方向以 JOG 模式移动。当运动因相关限位开关停止时，成功完成。	HomingDef[2]： JOG 速度。符号是方向，也定义了要查找的限位开关。 HomingDef[3]： 加速/减速。 HomingDef[4]： 紧急减速。 HomingDef[5]： 最大步骤时间，以[样本]为单位
检查是否超出限制	检查 RLS 和 FLS 是否未激活。如果其中一个被激活，则以适当的错误终止回零过程。	无参数
相对 PTP（PTP）移动	使用提供的运动参数移动到给定的相对距离。	HomingDef[2]：最大速度。 HomingDef[3]：加速/减速。 HomingDef[4]：相对距离（可以是正数或负数）。 HomingDef[5]：最大步骤时间，以[样本]为单位。

步骤类型	步骤描述	HomingDef[2-10]的步骤参数 （或[12-20]，[22-30] …）
慢速移动到索引	使用提供的运动参数慢速移动，直到检测到索引。慢速移动的速度应足够低，以确保检测到索引。理论上，速度应低于 16384 计数/秒。推荐值小于 8000 计数/秒。	HomingDef[2]：慢速移动速度。符号表示运动方向。 HomingDef[3]：加速/减速。 HomingDef[4]：紧急减速。 HomingDef[5]：最大步骤时间，以[样本]为单位。
移动到索引位置	使用提供的运动参数移动到最后记录的索引位置。	HomingDef[2]：最大速度。 HomingDef[3]：加速/减速。 HomingDef[4]：紧急减速。 HomingDef[5]：最大步骤时间，以[样本]为单位。
设置位置	将当前位置设置为提供的值。 请注意，如果不满足 SetPosition 的条件，此步骤将不执行任何操作。请参阅 SetPosition 关键字手册页。	HomingDef[2]：要设置为当前位置的期望位置值。
等待 N 个样本	等待 N（由步骤参数提供）个样本。	HomingDef[2]：在进入下一步之前等待的周期数。

步骤类型	步骤描述	HomingDef[2-10]的步骤参数 （或[12-20]，[22-30] …）
启用（或禁用）电机	启用或禁用电机	HomingDef[2]：0 表示禁用电机。 1 表示启用电机。 HomingDef[3]：最大步骤时间，以[样本]为单位。
移动到硬停止（通过电机堵转检测）	使用提供的运动参数移动，直到通过提供的参数检测到电机堵转。 当电机速度（绝对值）低于给定的速度阈值且电机电流（绝对值）高于给定的电机电流阈值时，检测到电机堵转，且在给定的堵转时间内连续。 一旦检测到电机堵转，电机位置将设置为给定的“设置位置”值。 最大速度参数的符号定义了运动方向。 请注意，如果不满足 SetPosition 的条件，此步骤将不设置期望位置。请参阅 SetPosition 关键字手册页。	HomingDef[2]：最大速度。 HomingDef[3]：加速/减速。 HomingDef[4]：紧急减速。 HomingDef[5]：检测电机堵转的速度阈值。 HomingDef[6]：电机电流（以 mA 为单位）阈值，用于检测电机堵转。 HomingDef[7]：堵转时间（以样本为单位）。 HomingDef[8]：检测到硬停止时要设置的位置。 HomingDef[9]：最大步骤时间，以[样本]为单位。

步骤类型	步骤描述	HomingDef[2-10]的步骤参数 （或[12-20]，[22-30] …）
移动到硬停止（通过高位置误差检测）	使用提供的运动参数移动，直到通过提供的参数检测到高位置误差来检测硬停止。 当位置误差（绝对值）高于提供的最大位置误差阈值时，检测到电机堵转。 一旦检测到电机堵转，电机位置将设置为给定的“要设置的位置”值。 最大速度参数的符号定义了运动的方向。 请注意，如果不满足 SetPosition 的条件，此步骤将不会设置所需位置。 请参阅 SetPosition 关键字手册页。	HomingDef[2]：最大速度。 HomingDef[3]：加速/减速。 HomingDef[4]：紧急减速。 HomingDef[5]：检测电机堵转的最大位置误差阈值。 HomingDef[6]：检测到硬停止时要设置的位置。 HomingDef[7]：最大步骤时间，以[样本]为单位。

步骤类型	步骤描述	HomingDef[2-10]的步骤参数 （或[12-20]，[22-30] …）
JOG 到 Home 离散输入的变化	使用提供的运动参数移动，直到检测到 Home 离散输入的变化。一旦检测到，停止运动。 给定最大速度参数的符号和 Home 离散输入的状态定义了运动的方向。 如果 Home 离散输入为“0”，则最大速度参数的符号为运动方向。如果 Home 为“1”，则方向相反。	HomingDef[2]：最大速度。 HomingDef[3]：加速/减速。 HomingDef[4]：紧急减速。 HomingDef[5]：最大步骤时间，以[样本]为单位。
绝对 PTP（PTP）移动	使用提供的运动参数移动到给定的绝对目标位置。	HomingDef[2]：最大速度。 HomingDef[3]：加速/减速。 HomingDef[4]：绝对目标位置。 HomingDef[5]：最大步骤时间，以[样本]为单位。

步骤类型	步骤描述	HomingDef[2-10]的步骤参数 （或[12-20]，[22-30] …）
设置软件位置限制	提供可选设置反向软件位置限制（RevPLim 参数）值的方法，同样也适用于前向限制。	HomingDef[2]：设置 RevPLim 值？1 为设置，0 为不设置。 HomingDef[3]：RevPLim 的新值（如果 HomingDef[2]为 0 则忽略）。 HomingDef[4]：设置 FwdPLim 值？1 为设置，0 为不设置。 HomingDef[5]：FwdPLim 的新值（如果 HomingDef[4]为 0 则忽略）。
配置位置锁定	配置位置锁定功能	HomingDef[2]：0 为禁用锁定，1 为启用锁定。类似于 LockEn。 HomingDef[3]：定义锁定源和极性。类似于 LockSrc。 HomingDef[4]：最大步骤时间，以[样本]为单位。
JOG 到锁定（通过硬件索引）	使用提供的运动参数 JOG，直到发生锁定并锁定位置被锁存。然后减速并停止。	HomingDef[2]：JOG 速度。符号为运动方向。 HomingDef[3]：加速/减速。 HomingDef[4]：紧急减速。 HomingDef[5]：最大步骤时间，以[样本]为单位。

步骤类型	步骤描述	HomingDef[2-10]的步骤参数 （或[12-20]，[22-30] …）
移动到锁定位置	使用提供的运动参数移动到最后记录的锁定位置（LockVal）。	HomingDef[2]：最大速度。 HomingDef[3]：加速/减速。 HomingDef[4]：紧急减速。 HomingDef[5]：最大步骤时间，以[样本]为单位。

大多数步骤类型 包含 内置错误检测机制。例如，超时或达到意外的限位开关等。当在回零过程中检测到此类错误时，过程将中止，并且 HomingStat 将被正确设置以反映错误类型（详见上文）。

请注意，在进入回零过程时，控制器的运动参数（速度、加速度、减速度和紧急减速度） 会被临时保存，并在回零过程完成后恢复。这是必要的，因为回零过程可能会改变这些参数的值。

另请参阅：

HomingDef, HomingStat, MotionReason , SetPosition 和 ConFlt。

项目	内容
关键字（Mnemonic）	HomingDef
CAN 代码	341
类型	数组参数（索引范围 1..150）
访问权限	读/写
允许在运动中	是
允许伺服开启时	是
存储到闪存	是
是否与轴相关	是
范围	-2,147,483,648 到 2,147,483,647
默认值	0
用户单位	无（No user units）
实现状态	已实现

作用：

- 定义和存储多个不同的回零（Homing）策略。
- 每个元素代表特定的回零配置，包括步骤顺序、参数等。
- 详细配置可以涉及限制检测、位置设定、等待、启用/禁用电机等步骤。

参照：

- 更多详细信息请查阅 HomingOn 关键词页面，它描述了回零的整体流程和配置。

项目	内容
关键字（Mnemonic）	RetractSpeed
CAN 代码	608
类型	32 位整数（int）
访问权限	读/写
是否与轴相关	是
用户单位	以用户单位为准
允许在运动中	是
允许伺服开启时	是
存储到闪存	是
范围	-1,300,000,000 到 1,300,000,000
默认值	1,000

作用：

- 控制用于“退回操作”的速度参数。
- 通常用于返回（Rebtraction）或反向操作中的速度设置，特别是在力控控制（force control）中作为默认速度。

RetractTarget（退回目标位置）

项目	内容
关键字（Mnemonic）	RetractTarget
CAN 代码	609
类型	32 位整数（int）
访问权限	读/写
是否与轴相关	是
用户单位	以用户单位为准
允许在运动中	是
允许伺服开启时	是
存储到闪存	是
范围	-2,147,483,648 到 2,147,483,647

项目	内容
默认值	0

作用：

- 设置在力控控制（force control）模式中，退回操作的目标位置。
- 这个位置代表设备反向退回或校准时的目标点。

项目	内容
关键字（Mnemonic）	CurrDir
CAN 代码	76
类型	32 位整数（int）
访问权限	读/写
是否与轴相关	是
用户单位	无（No user units）
允许在运动中	否
允许伺服开启时	否
存储到闪存	是
范围	0（正方向）或 1（反方向）
默认值	0

作用：

- 控制电流（或磁极）方向。
- 只在设置或调试阶段可以修改，运动中不可变更。
- 常用于校准或调试，以定义运动的正反方向。

应用场景：

- 调整电机磁极方向，以确保运动方向正确。

项目	内容
关键字（Mnemonic）	RevPLim
CAN 代码	82
类型	32 位整数（int）
访问权限	读/写
是否与轴相关	是
用户单位	以用户单位计
允许在运动中	否
允许伺服开启时	是
存储到闪存	是
范围	-2,147,483,648 到 2,147,483,647
默认值	-2,000,000,000

作用：

- 定义电机的反向位置限制（最小位置）。
- 若当前位置超过该限制，电机将停止运动，不能在负方向启动新运动。
- 只有当前位置返回到限制范围内，才允许正向运动。

相关参数：

- **FWDPlim**：正向位置限制，配合使用定义运动范围。

项目	内容
关键字（Mnemonic）	FwdPLim
CAN 代码	83
类型	32 位整数（int）
访问权限	读/写
是否与轴相关	是
用户单位	以用户单位计
允许在运动中	否
允许伺服开启时	是
存储到闪存	是
范围	-2,147,483,648 到 2,147,483,647
默认值	2,000,000,000

作用：

- 定义电机的正向最大位置限制。
- 当当前位置超出该限制，电机会停止，不能启动新的正向运动。
- 只有当前位置返回到限制范围内，才允许负向运动。

相关参数：

- 结合 RevPLim 共同定义运动范围，确保运动安全。

项目	内容
关键字（Mnemonic）	MaxPosErr
CAN 代码	84
类型	32 位整数（int）
访问权限	读/写
是否与轴相关	是
用户单位	以用户单位为准
允许在运动中	是
允许伺服开启时	是
存储到闪存	是
范围	0 到 80,000,000
默认值	20

作用：

- 定义允许的最大位置偏差（误差）。
- 位置误差是目标位置（PosRef）与实际位置（Pos）之间的差值。
- 超过此误差，电机会被禁用，作为失效保护措施。

项目	内容
关键字（Mnemonic）	MaxVelErr
CAN 代码	85
类型	32 位整数（int）
访问权限	读/写
是否与轴相关	是
用户单位	以用户单位为准
允许在运动中	是
允许伺服开启时	是
存储到闪存	是
范围	0 到 1,300,000,000
默认值	32,768

作用：

- 控制允许的最大速度误差。
- 如果实际速度误差（目标速度与实际速度差异）超过此值，电机会被禁用，起到安全保护作用。

项目	内容
关键字（Mnemonic）	HomingStep
CAN 代码	385
类型	32 位整数（int）
访问权限	只读
是否与轴相关	是
用户单位	无（No user units）
允许在运动中	是
允许伺服开启时	是
存储到闪存	否
范围	-2,147,483,648 到 2,147,483,647
默认值	0

作用：

- 表示一次回零操作中的具体步骤编号或当前步骤。
- 作为只读参数，用于监控和诊断回零过程的状态。

应用场景：

- 在自动回零流程中，用于追踪回零步骤状态。
- 不允许直接修改，由系统控制。

ComtMode（回零模式）

项目	内容
关键字（Mnemonic）	ComtMode
CAN 代码	72
类型	参数（数组，索引 1 至 24）
访问权限	读写
是否与轴相关	是
用户单位	无（No user units）
允许在运动中	否
允许伺服开启时	否
存储到闪存	是
范围	-2,147,483,648 到 2,147,483,647

索引详解

索引	描述	取值示例/说明
1	自动相位方法	• 0：跳转至零相位 • 1：保留 • 2：带预定义偏移的绝对编码器 • 3：带 Halls 切换的特殊编码器 • 4：带 Halls/编码器切换的编码器 • 5：最小跳跃搜索 • 6：仅 Halls 换向
2	定义跳转零位方法的电压	用于自动相位的电压参数
3	定义跳转零位方法的持续时间	用于自动相位的持续时间参数
4	绝对编码器自动相位中，零电角时的编码器读数	参考值
5	开始自动相位的触发值（1282）	当此值被设置时，触发自动相位开始
6	跳转至零点自动相位的过程配置	0：使用跳跃；1：使用平滑过程（推荐）
7-16	保留	未来扩展预留空间
18	自动相位结束的精度要求	设定误差容许范围
19	自动相位的触发事件	• 0：开机自动 • 1：仅手动请求 • 2：自动开机（需要）

索引	描述	取值示例/说明
		• 3：开机和 MotorOn 都自动
20–23	最小跳跃方法的配置参数	用于优化自动相位的程序调节

用途和应用场景

- 设置自动回零和相位调节的详细参数。
- 控制自动回零行为、精度、触发条件等。
- 通过多索引参数实现不同的自动校准策略。

换向是通过在不同电机相位之间交替电流来产生运动的过程 （仅适用于无刷直流电机） 。为了正确换向电流，控制器必须获取电机的电气角度信息。此信息可以从霍尔传感器接收，例如 。

在没有霍尔传感器的情况下， 电机的电气角度必须从编码器读数中推导出来，这通常与电机的电气角度不对齐。 因此，需要一个初始化过程来对齐编码器读数和电机的电气相位。

这可以通过几种方法之一来完成。

ComtMode [ ] 是一个控制电气角度检测过程的数组。

ComtMode[1] 选择将用于检测电气角度的方法：

- 0 – 跳到零。电机将从当前位置跳到电气零 角度 的一个相位。此方法通常涉及较大的突然运动。
- 1 – 最小运动。在此方法中，电机将仅执行非常小的运动，并在其当前位置检测电气角度。
- 2 – 绝对编码器。对于绝对编码器，使用上述方法之一对每个电机+控制器组执行一次角度检测。检测到角度后，使用保存命令保存所有参数。角度 0 的绝对位置将保存在控制器的闪存中。在同一电机-控制器组的所有后续使用中，使用 ComtMode[1]=2 使用保存在内存中的数字计算当前电气角度，而无需任何运动。

使用“跳到零”方法时，用户可以确定在过程中将施加到电机上的 电压 的量。如果 电压（因此产生的电流） 太低，系统中的机械因素（如摩擦）可能会阻止电机移动并导致过程失败。长时间对同一相位施加高电流可能会造成损坏。用户应施加足够移动但不过多的电流。

电机的电压（因此电流）在一段时间内逐渐增加，因此最大电流仅在短时间内施加。 每个电压（电流）在 一个步骤 的持续时间内施加 10 毫秒。

ComtMode[2]：仅在“跳到零”模式下使用。确定在每个步骤后电机的电压增加多少。结果电流取决于总线电压和电机电阻。为了防止损坏，建议在第一次实验时将此数字设置得较低，并仅在必要时提高。如果需要更改 ComtMode[3]。

ComtMode[3] 确定在角度检测过程中将应用多少电压步骤。每个电压值施加到电机上 10 毫秒然后电压增加。建议从 1 秒检测过程开始（ ComtMode[3] = 100）。检测完成后，禁用电机，手动将电机移动到不同位置并重复该过程。电机应移动。如果您没有看到任何运动，请增加 ComtMode[3] 并再次移动电机，直到看到导致运动的检测过程。只要您没有看到运动，可能是电流不足以克服摩擦或电机最初就在 0 位置。

关于“最小运动”方法：

使用增量编码器（或首先使用绝对编码器的电机），在上电后没有关于电机电气角度的信息。“跳到零”方法通过强制电机跳到已知电气角度（0 度）来克服这一困难。但这涉及相对较大的电机跳跃。“最小运动”方法执行不同的过程，以最小化这种电机运动。

一般来说，想法是在过程中（大约 2-3 秒）暂时关闭一个控制回路，该回路试图保持电机在其当前位置，同时用固定电流命令（由用户设置）和控制回路的输出电气角度命令电流控制回路。结果是，控制回路（试图在过程开始时保持电机在其位置）将找到合适的电气角度，尽管电流施加到电机上，但仍能保持电机在此位置。

这就是电机位置的电气角度，减去 90 度。

因此，控制回路在其当前位置找到电机的电气角度。

由于最初不知道此角度（假设为零角度），电机将稍微移动，直到控制回路稳定到稳态，电机将回到其原始位置。

为了最佳克服摩擦并确保过程的准确性，在此阶段（闭环）完成后，将执行第二阶段。在此阶段，用户定义的电流（高于闭环中使用的电流）在开环中施加到电机上，以将其强制到与检测到的角度相关的位置。

知道检测到的角度及其相关位置，驱动器可以初始化换向变量（电气周期的“零”在哪里）并准备好运动。

使用“最小运动”方法时，用户应确定在过程中将使用的电流量，如下所示：

ComtMode[6]：在闭环中寻找电气角度的过程中使用的电流，以[mA]为单位，如上所述。 通常，它约为电机峰值电流的 10%到 20%。它必须高于克服系统摩擦所需的电流。

ComtMode[7]：在最小运动换向过程的最后阶段使用的电流，以[mA]为单位。在此阶段，电流在开环中施加，以便电机被强制到与检测到的电气角度相关的位置。 通常，它约为电机峰值电流的 15%到 30%。它必须高于克服系统摩擦所需的电流，并且为了获得最佳性能，它应高于 ComtMode[6]。

请注意，虽然 ComtMode[6] 和 [7] 用于在过程中设置电流参考（CurrRef），但 CurrRef 的实际值受用户定义的驱动电流限制。

由于“最小运动”方法涉及闭环控制以在当前位置寻找电气角度，因此它涉及闭环的增益和积分项参数，如上所述。用户应按如下方式设置这些参数：

ComtMode[8]：是 PI 控制滤波器的增益。典型值为 5000。根据系统、编码器分辨率和电机的极数可能需要更改。

ComtMode[9]：是 PI 控制滤波器的积分增益。典型值为 20。根据系统、编码器分辨率和电机的极数可能需要更改。

ComtMode[4] 保存具有绝对编码器的电机的电气零位置。如果需要更换电机或控制器，则必须重新检测电机的电气角度并保存。 ComtMode[4] 的值在使用 ComtMode[1]=0 或 1 进行换向过程后自动分配。 这意味着对于具有绝对编码器的电机，您应首先使用 ComtMode[1] = 0（或 1）执行检测过程，然后在过程成功完成并且控制器自动更新 ComtMode[4] 后，您可以更改为 ComtMode[1]=2（“绝对编码器”模式）并保存到闪存中，以便下次不需要任何运动。

ComtMode[5] 用于请求新的换向过程。要重复换向过程，请输入 ComtMode[5] = 1282。将开始新的换向过程，并清除 ComtMode[5] 的值。

换向过程在上电或重置后自动执行。

特定产品说明：

1. 产品 1：
    1. 目前，仅支持“跳到零”方法。
    2. 重复换向过程是通过 ComtMode[1]=1282（而不是 ComtMode[5]）。

ComtAng（换向角）

项目	内容
关键字（Mnemonic）	ComtAng
CAN 代码	73
类型	只读参数（32 位整数）
是否与轴相关	是
用户单位	无（No user units）
允许在运动中	是
允许伺服开启时	是
存储到闪存	否
范围	0 到 35,999
默认值	0

ComtAng 是编码器计数中的换向角，实际上是当前位置与电机实际电气零位置之间的编码器计数距离。换向角是电机在当前电气周期内的角度。

示例：

假设一个具有 4000 计数编码器的 2 对极电机。每个电气周期为 2000 计数。如果电机位置当前距离检测到的电气零位置为 500 计数，则 ComtAng = 500，这意味着电机距离电气零 90 度。

注意：

电气零位置定义为电机相位 A 的正弦反电动势（或扭矩/力）“常数”为 0 的位置。

项目	内容
关键字（Mnemonic）	RelTrgt
CAN 代码	135
类型	32 位整数（int）
访问权限	读写
是否与轴相关	是
用户单位	以用户单位计
允许在运动中	是
允许伺服开启时	是
存储到闪存	否
范围	-2,147,483,648 到 2,147,483,647
默认值	0

作用：

- 定义 PTP（点到点）和重复运动中的相对运动距离。
- 作为运动的目标位置，相对于当前位置的偏差。

详细说明：

- **PTP 运动**：代表从当前位置到目标位置的偏移。
- **重复运动**：定义停止位置，相对当前或初始位置。
- 如果 `RelTrgt ≠ 0`，则忽略 `AbsTrgt`，直接使用 `RelTrgt` 作为运动目标。
- 在运动过程中可以动态改变 `RelTrgt`，目标会立即调整。

示例：

1. 当前 Pos=1000，AbsTrgt=5000，RelTrgt=0：下一次 PTP 结束点为 5000。
2. 当前 Pos=1000，AbsTrgt=5000，RelTrgt=7000：下一次 PTP 结束点为 Pos = 1000 + 7000 = 8000。
3. 当前 Pos=1000，AbsTrgt=5000，RelTrgt=7000：连续重复运动将前往 Pos=8000，然后返回 Pos=1000，不断循环。

项目	内容
关键字（Mnemonic）	AbsTrgt
CAN 代码	134
类型	32 位整数（int）
访问权限	读写
是否与轴相关	是
用户单位	以用户单位计
允许在运动中	是
允许伺服开启时	是
存储到闪存	否
范围	−2, 147, 483, 648 到 2, 147, 483, 647
默认值	0

作用：

1. 定义 PTP（点到点）和重复运动的绝对目标位置。
2. 无论当前位置如何，运动结束所达到的目标位置。

详细说明：

1. **PTP 运动**：此为终点位置。
2. **重复运动**：定义机电停止的位置，连续运动时会往返于此位置与起始位置之间。
3. 如果 `RelTrgt ≠ 0`，则忽略 `AbsTrgt`，以 `RelTrgt` 作为实际目标。
4. 在运动过程中可以动态改变 `AbsTrgt`，目标会立即更新。

示例：

1. 当前 Pos=1000， AbsTrgt=5000，RelTrgt=0：下一次 PTP 在 Pos=5000 结束。
2. 当前 Pos=1000， AbsTrgt=5000，RelTrgt=7000：下一次 PTP 在 Pos=8000（1000 + 7000）。
3. 当前 Pos=1000，AbsTrgt=5000，RelTrgt=0：重复运动会到 Pos=5000，然后返回 Pos=1000，如反复循环。

项目	内容
关键字（Mnemonic）	RptWait
CAN 代码	147
类型	32 位整数（int）
访问权限	读写
是否与轴相关	是
用户单位	毫秒(ms)
允许在运动中	是
允许伺服开启时	是
存储到闪存	是
范围	0 到 131,071,993
默认值	0

作用：

- 在重复运动模式下，定义每个端点停留的时间（以毫秒为单位）。
- 支持调整电机在两个点之间来回运动的等待时间。

应用场景和意义：

- 方便调试：运动结束与下一次开始的时间分隔明显，便于观察和记录运动轨迹。
- 让系统在运动中减速、衰减，避免突发变化，便于监控。
- 长时间运动时，添加等待时间可以防止硬件过热。

使用建议：

- 在复杂或高加速度/减速度运动中，可以通过设置等待时间缓冲系统，保障运动平稳。

RptMode（重复模式）

项目	内容
关键字（Mnemonic）	RptMode
CAN 代码	712
类型	32 位整数（int）
访问权限	读写（可获取和设置）
是否与轴相关	是
用户单位	无（属于模式编号或代码）
默认值	712 （如果你的需求是设置为 712）

RptMode 定义运动的重复模式，即运动将以何种方式重复执行。常见的模式可能包括：

- **0**：无重复（单次运动）
- **1**：循环往返（往返运动不停重复）

RptCycles （执行的重复次数）

项目	内容
关键字（Mnemonic）	RptCycles
CAN 代码	713
类型	32 位整数 (int)
权限	读写
是否与轴相关	是
用户单位	次（次重复次数）
允许在运动	是
允许与电机开启时	是
存储到闪存	是
范围	0 到 2,147,483,647（最大值为 32 位整数最大值）
默认值	0

功能作用：

- RptCycles 用于设定在多次重复运动模式下，总共要执行的重复次数。
- 配合 RptMode 设置为“固定次数重复”时。

项目	内容
关键字（Mnemonic）	RptCounter
CAN 代码	714
类型	32 位整数（int）
权限	只读（不能修改，只能读取）
是否与轴相关	是
用户单位	次（已完成的重复次数）
允许在运动中	是
允许与电机开启时	是
存储到闪存	否
范围	0 到 2,147,483,647（最大值为 32 位整数最大值）
默认值	0（运动开始前默认值）

功能作用：

- RptCounter 用于实时读取已完成的重复次数。

CurrPosTh （切换力控位置阈值）

项目	内容
关键字（Mnemonic）	CurrPosTh
CAN 代码	426
类型	32 位整数 (int)
权限	读写
是否与轴相关	是
用户单位	用户定义单位（可能是位置单位）
允许在运动中	是
允许与电机开启时	是
存储到闪存	是
范围	-2,147,483,648 到 2,147,483,647
默认值	0

功能作用：

必须满足的全局阈值,以便切换到电流或力操作模式。 如果位置参考（PosRef)大于或小于阈值，则可以进行切换（如果触发了其他任何阈值）

CurrPosThDir（切换力控位置阈值方向）

项目	内容
关键字（Mnemonic）	CurrPosThDir
CAN 代码	427
类型	32 位整数（int）
权限	读写
是否与轴相关	是
用户单位	无（仅为方向标志，即方向参数）
允许在运动中	是
允许与电机开启时	是
存储到闪存	否
范围	-1 到 1
默认值	0

功能作用：

- CurrPosThDir 表示当前位置阈值的**方向**，只取值为 -1, 0, 1。
  - -1：意味着阈值在当前位置的反方向。
  - 0：无特定方向设置，默认行为（通常代表不偏向任何方向）。
  - 1：阈值在当前位置的正方向。

项目	内容
关键字（Mnemonic）	BeginOnToPos
CAN 代码	587
类型	32 位整数 (int)
权限	读写
是否与轴相关	是
用户单位	无（0 或 1）代表布尔值，控制是否自动开始运动
允许在运动中	是
允许与电机开启时	是
存储到闪存	否
范围	0（不自动开始）至 1（自动开始）
默认值	0

用于控制器在切换到位置模式并回退到设定位置的行为：

- **值为 1**：自动开始运动（在模式切换时自动启动运动，方便快速连续操作）
- **值为 0**：不启动运动

应用场景：

- 在力控模式或位置控制切换场景下，启用此参数可以开启回退功能，实现自动启动。

16.2 矢量关键字

VecAbsTrgt（矢量运动目标位置）

项目	内容
关键字（Mnemonic）	VecAbsTrgt
CAN 代码	642
类型	只读参数（Read only）
权限	只读（不能设置）
是否与轴相关	是
允许在运动中	是
允许与电机开启时	是
存储到闪存	否
值范围	-2,147,483,648 至 2,147,483,647
默认值	0
用户单位	无（无用户单位，直接数值）

功能说明

- `VecAbsTrgt` 表示当前矢量运动的目标距离（从起点到终点），只是一个状态参数，用于监控运动距离。
- **它不是用来定义运动路径**，真正的运动定义由相关的 `RelTrgt`（相对目标距离）或成员轴的 `AbsTrgt` 定义。
- **注意**：`VecAbsTrgt` 总是正值，表示绝对距离。

VecAccel（矢量运动加速度）

项目	内容
关键字（Mnemonic）	VecAccel
CAN 代码	636
类型	读写参数（Read / Write）
权限	可以在运动中设置
是否与轴相关	是
存储到闪存	是
允许在运动中	是
允许与电机开启时	是
值范围	100 到 2,000,000,000
默认值	100,000
用户单位	用户单位（对应实际应用的运动单位）

功能说明

- VecAccel 设置矢量运动中的加速度。数值越大，加速度越快。
- 该参数在运动开始或运动中动态调整都有效。
- 具体数值应根据硬件能力和运动需求合理设置。

VecArcCenter （矢量运动弧形运动中心点）

项目	内容
关键字（Mnemonic）	VecArcCenter
CAN 代码	633
类型	参数（Parameter）
权限	读写（Read / Write）
允许在运动中	否（运动时不能修改）
允许与电机开启时	是
存储到闪存	是
是否与轴相关	是
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0
用户单位	以用户单位为准

功能说明

- `VecArcCenter` 定义弧线运动中弧的中心位置。
- 适用于弧线运动（Arc Vectors），通过设置弧心坐标，控制器可以计算出半径。
- 该参数与所有其他矢量运动球参数配合使用，确保运动轨迹的正确性。
- 只在非运动状态下可修改（即不能在运动时改变该参数）。
- **注意：**每个轴的 `VecArcCenter` 定义了弧心的坐标。

相关参数参考

- `VecAccel`：运动加速度
- `VecAbsTrgt`：目标绝对距离
- `VecArcDir`：弧线方向
- `VecEncRatio`：编码器比例

项目	内容
关键字（Mnemonic）	VecArcDir
CAN 代码	634
类型	参数（Parameter）
权限	读写（Read / Write）
允许在运动中	否（运动时不能修改）
允许与电机开启时	是
存储到闪存	是
是否与轴相关	是
最小值	0
最大值	1
默认值	0
用户单位	无（没有用户单位）

功能说明

VecArcDir 定义弧线运动的方向：

"0" 表示逆时针（CCW）

"1" 表示顺时针（CW）

只在使用 Begin 命令时依据该参数的 VecArcCenter 进行运动。

两个轴定义来进行弧线运动：

运动的平面由两个轴定义：第一个轴为 X 轴，第二个轴为 Y 轴。

弧线在这两个轴的平面内，第三轴不变。

顺序很重要：

例如：B,C 与 C,B，不同的顺序代表不同的弧方向。

默认第一个轴为 X 轴，第二个轴为 Y 轴，弧的运动方向由 VecArcDir 决定。

注意：

运动定义在运动前设置完毕后不可修改。

设计需要与 CNC 的弧定义保持一致，以确保运动准确。

相关参数参考

- VecAccel：运动加速度
- VecAbsTrgt：目标距离
- VecArcCenter：弧心位置
- VecEncRatio：编码器比例
- 其他事件参数：oEventOn、oEventGap 等

VecDecel （矢量运动减速度）

项目	内容
关键字（Mnemonic）	VecDecel
CAN 代码	637
类型	读写参数（Read / Write）
允许在运动中	是
允许与电机开启时	是
存储到闪存	是
是否与轴相关	是
最小值	100
最大值	2,000,000,000
默认值	100,000
用户单位	用户单位（与 VecAccel 一致）

功能说明

- VecDecel 定义矢量运动中的减速度，其数值越大，减速越快。
- 运动过程中可以动态调整。
- 与 VecAccel 配合使用，控制运动的加减速特性。

相关参数参考

- VecAccel：加速度
- VecAbsTrgt：目标绝对距离
- VecArcCenter：弧心位置
- VecArcDir：弧线方向
- VecEncRatio：编码器比例

VecdPosRef（矢量运动位置的导数）

项目	内容
关键字（Mnemonic）	VecdPosRef
CAN 代码	644
类型	只读参数（Read only）
允许在运动中	是
允许与电机开启时	是
存储到闪存	否
是否与轴相关	是
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0
用户单位	无（数值无用户单位，表示为导数速率）

功能说明

- VecdPosRef 表示矢量运动参考位置的导数，即运动速度的数值。
- 它是一个状态参数，始终为正（代表速度大小），用以监控运动瞬间的速度情况。
- 在运动中可以实时读取。

VecEmrgDec（矢量运动紧急减速度）

项目	内容
关键字（Mnemonic）	VecEmrgDec
CAN 代码	638
类型	读写参数（Read / Write）
允许在运动中	是
允许与电机开启时	是
存储到闪存	是
是否与轴相关	是
最小值	100
最大值	2,000,000,000
默认值	100,000
用户单位	用户单位（对应运动的实际单位）

功能说明

- VecEmrgDec 定义运动的应急减速度，数值越大减速越快。
- 在突发紧急停止或其他需要快速减速的情况下，控制器会使用此参数的值，确保安全响应。

相关参数

- VecAccel：加速度
- VecDecel：减速度
- VecAbsTrgt：目标距离
- VecArcCenter、VecArcDir：弧运动参数
- VecEncRatio：编码器比例

VecEncRatio （矢量运动编码器分辨率）

项目	内容
关键字（Mnemonic）	VecEncRatio
CAN 代码	632
类型	读写参数（Read / Write）
允许在运动中	否（运动中不能修改）
允许与电机开启时	是
存储到闪存	是
是否与轴相关	是
最小值	256
最大值	25,600
默认值	256（比例=1）
用户单位	无（数值为比例系数）

功能说明

- VecEncRatio 用于补偿每个轴的编码器分辨率差异。
- 控制器内部实际比例为其值除以 256，例如：
- VecEncRatio=256 表示比例为 1
- VecEncRatio=260 表示比例为 260/256 ≈ 1.016
- 使得运动位置与编码器读数同步，确保运动的精度。
- 在运动中禁止修改，以避免误差累积。

相关参数

- VecAccel：运动加速度
- VecDecel：减速度
- VecAbsTrgt：目标距离
- VecArcCenter、VecArcDir：弧运动参数
- VecEncRatio：编码器比率

VecJerk（矢量运动平滑系数）

项目	内容
关键字（Mnemonic）	VecJerk
CAN 代码	639
类型	读写参数（Read / Write）
允许在运动中	否（运动中不能修改）
允许与电机开启时	是
存储到闪存	是
是否与轴相关	是
最小值	0
最大值	9
默认值	0
用户单位	无（数值范围）

功能说明

- VecJerk 定义运动中跃变（Jerk）参数，用于控制运动的平滑性。
- 越大的值，运动变化越快，但可能会引入震荡或不平滑。
- 仅能在运动开始前设置，运动过程中不能修改。

相关参数参考

- VecAccel：加速度
- VecDecel：减速度
- VecJerk：跃变（Jerk）
- VecAbsTrgt：目标距离
- **运动参数中的跃变值，与运动平滑性直接相关**

VecMemberAxes （矢量运动参与轴）

项目	内容
关键字（Mnemonic）	VecMemberAxes
CAN 代码	631
类型	读写参数（Read / Write）
允许在运动中	否（运动中不能修改）
允许与电机开启时	是
存储到闪存	是
是否与轴相关	是
最小值	0
最大值	255
默认值	0
用户单位	无

功能说明

VecMemberAxes 使用 8 位二进制位来表示不同轴是否参与矢量运动。例如：

- bit 0 代表轴 A（第一轴）
- bit 1 代表轴 B（第二轴）
- 依此类推

通过设置不同的位值，可以组合多个轴参与运动。如：

- 选择轴 A 和轴 B：1 | 2 = 3
- 仅轴 A：1
- 仅轴 B：2

示例：
设置参与轴 A 和 B：VecMemberAxes = 3（二进制 00000011）

相关参数

- VecAccel：加速度
- VecDecel：减速度
- VecJerk：跃变
- VecAbsTrgt：目标位置
- VecArcCenter：弧心坐标
- 其他运动参数依赖于参与轴设置

VecMotionStat（矢量运动状态）

项目	内容
关键字（Mnemonic）	VecMotionStat
CAN 代码	641
类型	只读参数（Read only）
允许在运动中	是
允许与电机开启时	是
存储到闪存	否
是否与轴相关	是
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0
用户单位	无

功能说明

- VecMotionStat 用于获取矢量运动的当前状态，可以动态读取运动的状态信息。
- 由于为只读参数，不能修改，只用于监控和反馈。
- 具体状态编码请参考设备的状态码定义表（通常由多种标志或状态码组成，用于表示运动进行中、暂停、异常等状态）。

VecPause （矢量运动暂停）

项目	内容
关键字（Mnemonic）	VecPause
CAN 代码	640
类型	读写参数（Read / Write）
允许在运动中	是
允许与电机开启时	是
存储到闪存	否
是否与轴相关	是
最小值	0
最大值	1
默认值	0
用户单位	无

功能说明

- **值为 1 时：**暂停矢量运动，运动将速度降至 0，直到重新开始。
- **值为 0 时：**恢复正常运动（如果之前暂停，运动会加速到原设定速度 VecSpeed）。

VecPosRef （矢量位置参考值）

项目	内容
关键字（Mnemonic）	VecPosRef
CAN 代码	643
类型	只读参数（Read only）
允许在运动中	是
允许与电机开启时	是
存储到闪存	否
是否与轴相关	是
最小值	-2,147,483,648
最大值	2,147,483,647
默认值	0
用户单位	无

功能说明

- VecPosRef 表示当前矢量运动的实时位置参考值（沿路径的当前位置）。
- 它从 0 开始，运动结束后达到目标位置 VecAbsTrgt。
- **始终为正值**，反映运动的当前位置。
- 常用于监控运动进度。

VecSpeed （矢量运动速度）

项目	内容
关键字（Mnemonic）	VecSpeed
CAN 代码	635
类型	读写参数（Read / Write）
允许在运动中	是
允许与电机开启时	是
存储到闪存	是
是否与轴相关	是
最小值	0
最大值	60,000,000
默认值	10,000
用户单位	用户单位（具体单位依据运动设定）

功能说明

- VecSpeed 用于设定运动的目标速度，影响运动时的加速度决定及运动平滑性。
- 数值越大，运动速度越快。
- 可在运动过程中调整。

相关参数

- VecAccel：加速度
- VecDecel：减速度
- VecAbsTrgt：目标位置
- VecEncRatio：编码器比例
- VecJerk：跃变
- VecMotionStat：运动状态监控

项目	内容
关键字（Mnemonic）	VecType
CAN 代码	630
类型	读写参数（Read / Write）
允许在运动中	否
允许与电机开启时	是
存储到闪存	是
是否与轴相关	是
最小值	0
最大值	1
默认值	0
用户单位	无

功能说明

VecType = 0：直线运动。

VecType = 1：弧线运动。

未来可能扩展支持：

VecType = 2：复合弧线与直线运动（结合主运动弧和其他轴线的线性运动）。

不能在运动过程中修改，以保证运动路径和运动精度。

相关参数参考

- VecAccel：加速度
- VecAbsTrgt：目标位置
- VecArcCenter：弧心坐标
- VecArcDir：弧线方向
- VecSpeed：运动速度
- VecEncRatio：编码器比例

VecNumCircles （矢量运动附加圈数）

项目	内容
关键字（Mnemonic）	VecNumCircles
CAN 代码	646
类型	32 位整数（int）
访问权限	读写（Read / Write）
是否与轴相关	是
允许在运动中	否（运动中不能修改）
允许与电机开启时	是
存储到闪存	是
最小值	0
最大值	100
默认值	0

功能说明

- VecNumCircles 主要用于定义弧线或圆弧路径中的圆圈数，因为运动路径可以由多个弧圈组成，每个“圈”代表一段完整的弧。
- 设置为 0 意味着没有弧圈路径或仅为直线。

相关参数

- VecArcCenter：弧心坐标
- VecArcDir：弧线方向
- VecType：路径类型（线或弧）
- VecAbsTrgt：终点位置
- VecSpeed：运动速度

VecPosFOn （矢量运动位置滤波器开关）

项目	内容
关键字（Mnemonic）	VecPosFOn
CAN 代码	648
类型	32 位整数（int）
访问权限	读写（Read / Write）
是否与轴相关	是
允许在运动中	否
允许与电机开启时	是
存储到闪存	是
最小值	0
最大值	1
默认值	0

功能说明

- VecPosFOn 控制是否启用位置反馈：
  - **0：关闭**（默认）
  - **1：开启**
- 只可在运动停止时设置，不能在运动中修改。
- 主要用于启用或禁用运动过程中位置反馈的功能。

VecPosFDef（矢量运动位置滤波类型）

项目	内容
关键字（Mnemonic）	VecPosFDef
CAN 代码	647
类型	32 位整数数组（范围：1 到 5）
访问权限	读写（Read / Write）
是否与轴相关	是
允许在运动中	否
允许与电机开启时	是
存储到闪存	是
值范围	-2,147,483,648 到 2,147,483,647
默认值	0

功能说明

- VecPosFDef 是一个数组，索引范围为 1 到 5，用于预设不同方向或不同阶段的位置反馈默认值。

作用主要在设置或复位位置反馈的目标值或参考值。

- 无（0 - 无）：无滤波效果。
- 一级通用滤波器：基础一级滤波，用于简单滤波需求。
- 二级低通滤波器：较平滑的低通滤波，可以去除高频噪声。
- 一级超前后滤波器：在滤波同时引入相位超前，以补偿系统延时。
- 一级超后滤波器：在滤波后引入超前效果，改善系统响应。
- 二级超前后滤波器类型：更复杂的超前滤波器，结合多级滤波和超前功能。
- 二级超后滤波器类型：类似上面，但具有更高级的响应调整。
- 陷波（或陷波）：特定频率范围的滤波，用于抑制某个频率成分。
- 复杂超前后滤波器：复合滤波器，用于极为复杂的滤波和相位调整需求。

17 附录

通过以上文档，您可以详细了解 Agito-AAMotion 的使用方法及其功能。希望对您的项目有所帮助，如有其他疑问，欢迎查阅库的源码或相关文档。