Central-I 总线型　　直流驱控一体型　　交流驱控一体型　　运动控制器

Agito 产品系列

Agito 搭配 Twincat 的 Ethercat 力控手册 v1.0

agito

www.agito-akribis.com

Member of Akribis Systems group

版本记录

版本	描述	日期
1.0	首版发布	2024/12/12

※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

1 介绍

1.1 关于手册

感谢您选择 Agito 系列运动控制产品，我们将竭力为您提供追求速度与精度的极致运动控制方案，并提供全方位的技术支持。

本手册主要介绍 Agito 运动控制器在 Ethercat 通讯协议下的开环力控的使用方法。

手册中仅详细介绍与使用 Ethercat 通讯协议的开环力控配置内容，其他参数设置请参阅《Agito EC 驱动器连接 EtherCAT 主站参考手册 v1.0》中的详细介绍，本文档将不再累述。

1.2 内容简介

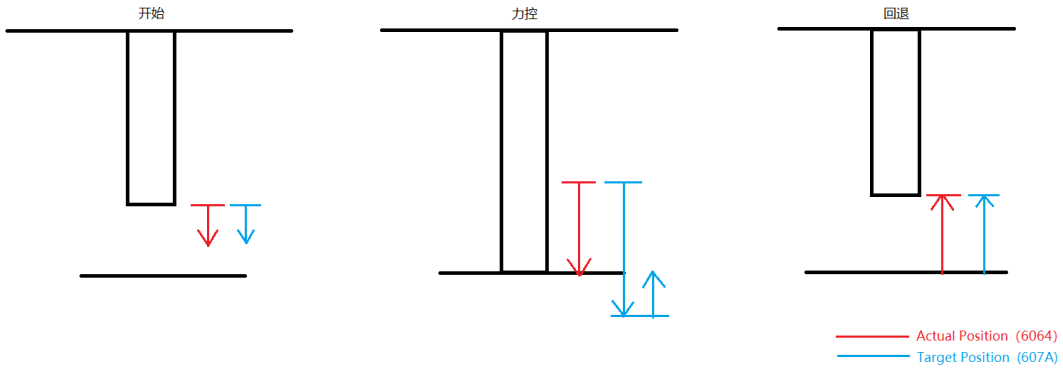
本手册是介绍在 Ethercat 通讯协议下控制器实现对电机的开环力控的方法。目前已经将 AGendata 写入 PDO 中，通过 PLC 程序对 PDO 中的 AGendata 修改实现 PLC 程序与 IDE 程序配合实现开环力控流程。本手册会分别介绍在 CSP 模式以及 PP 模式下的实现方法。

1.3 前言

不同主站需要各家不同的软件进行配置，进行开环力控相关步骤之前要确保已经配置好，可以进行基本的运动。手册会以在 Twincat 中配置为例介绍相关流程。

2 CSP 模式实现力控流程介绍

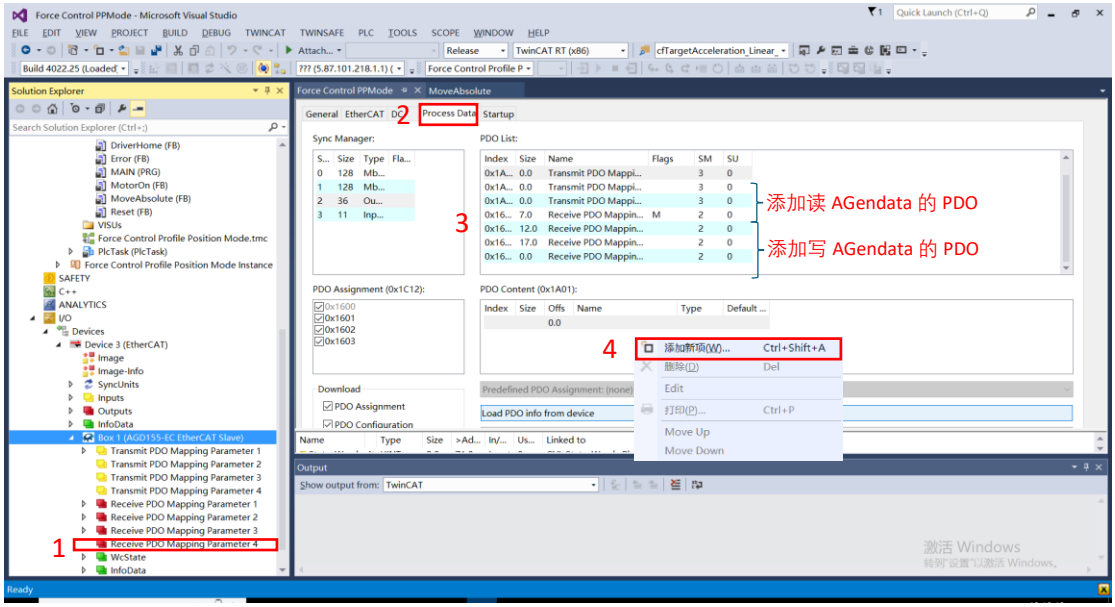
PLC 程序中会负责电机使能，复位，运动，报错等一系列内容，IDE 程序负责电机操作模式的切换以及力控结束的回退部分。PLC 程序发送使能指令后发送 AGendata 使 IDE 对一系列力控参数进行赋值，等待电机满足切换力控条件时切换为力控模式。后 PLC 程序再发送运动指令让电机运动到按压被压物体的位置。在执行力控期间，从站不会根据主站发出的命令进行运动，但是 Target position 会提前运动到该位置。我们在从站执行力控期间让主站运动到回退位置，等待电机执行完力控回退到目标位，两个位置重合后切换 Motionmode 为 19 后再进行后续运动。



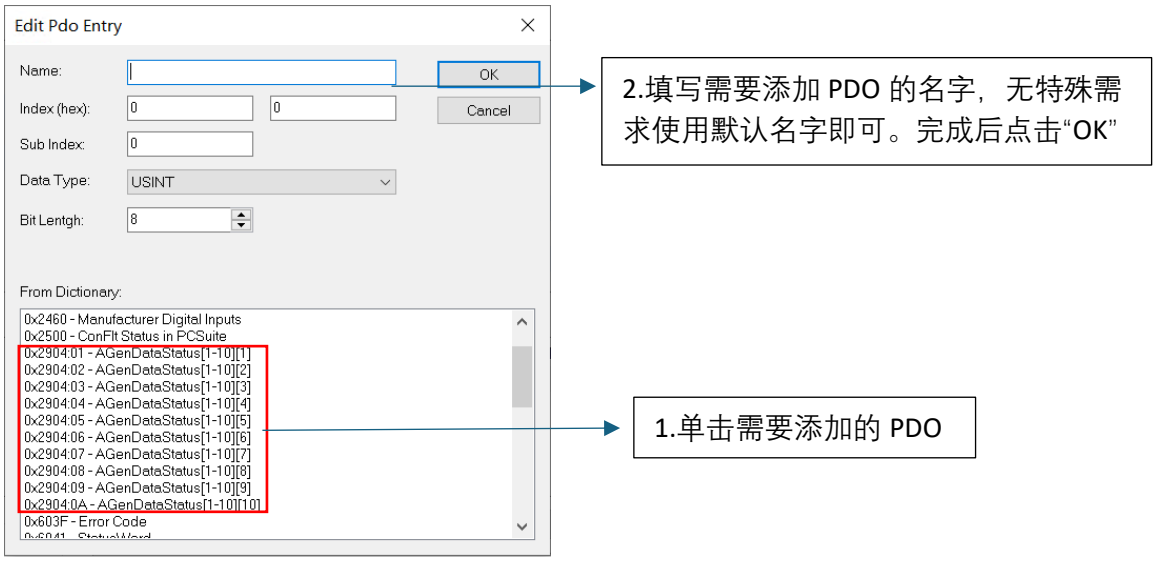
2.1 CSP 模式下力控流程操作步骤

2.1.1PLC 中读写 AGendata 的方法

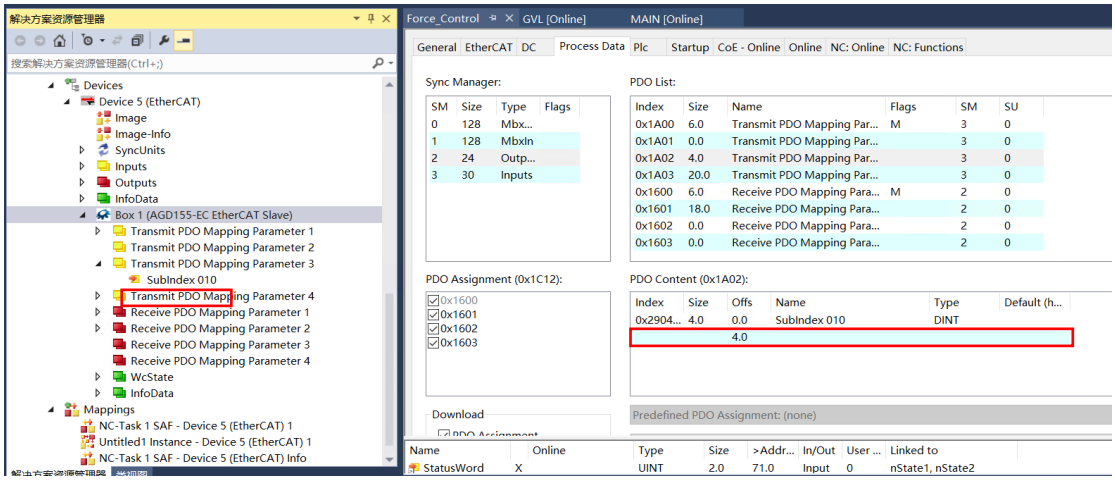
要使 PLC 程序和 IDE 程序搭配起来需要 PLC 能够读写 AGendata（控制器内部存储数据数组），通过读取不同 AGendata 的值触发 PLC 和 IDE 各自部分的程序。



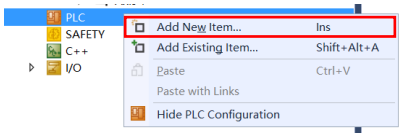
首先点击在 Device 中的从站 Box1，打开右侧的界面。打开负责 PDO 配置的 Process Data 的界面。右侧 PDO List 中在名字为 Transmit PDO Mapping Parameter 中添加读 AGendata 值的 PDO，在 Receive PDO Mapping Parameter 中添加写 AGendata 的 PDO。在下面的 PDO Content 中右击鼠标打开菜单栏点击“添加新项”打开下图界面。



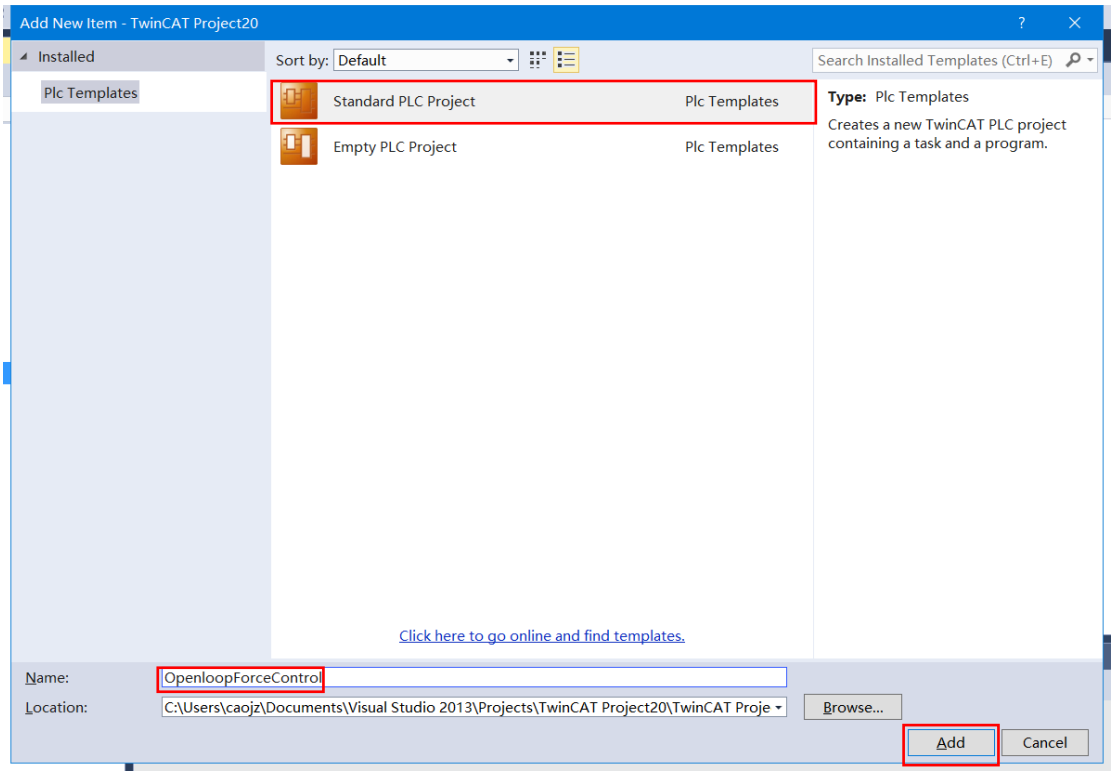
图中是 PLC 中可以读到 AGendata 的值 1-10 共十个。添加成功后如下图。



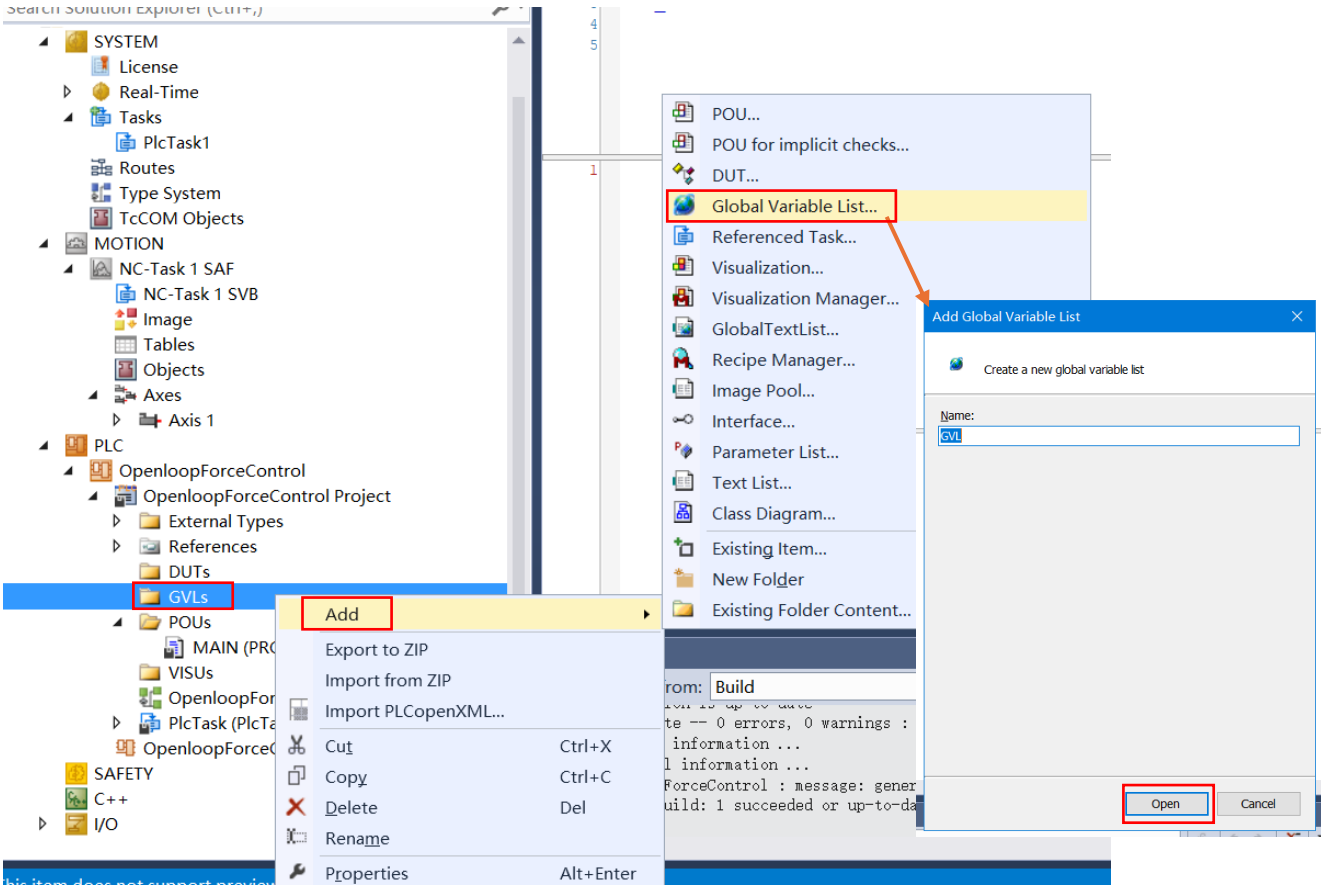
添加相关 PDO 后需要在 PLC 中定义全局变量，将全局变量和 PDO 关联起来后实现 PLC 程序读写 AGendata。



在 PLC 一栏添加新的项目。



选择 Standard PLC Project，命好名之后点击 Add 即可。



展开项目栏，在 GVLs 文件夹下添加 Global Variable List，命好名

后点击“Open”

```
{attribute 'qualified_only'}
VAR_GLOBAL

    AGENDATAR10                AT %I* : DINT;
    AGENDATA10                 AT %Q* : DINT;

END_VAR
```

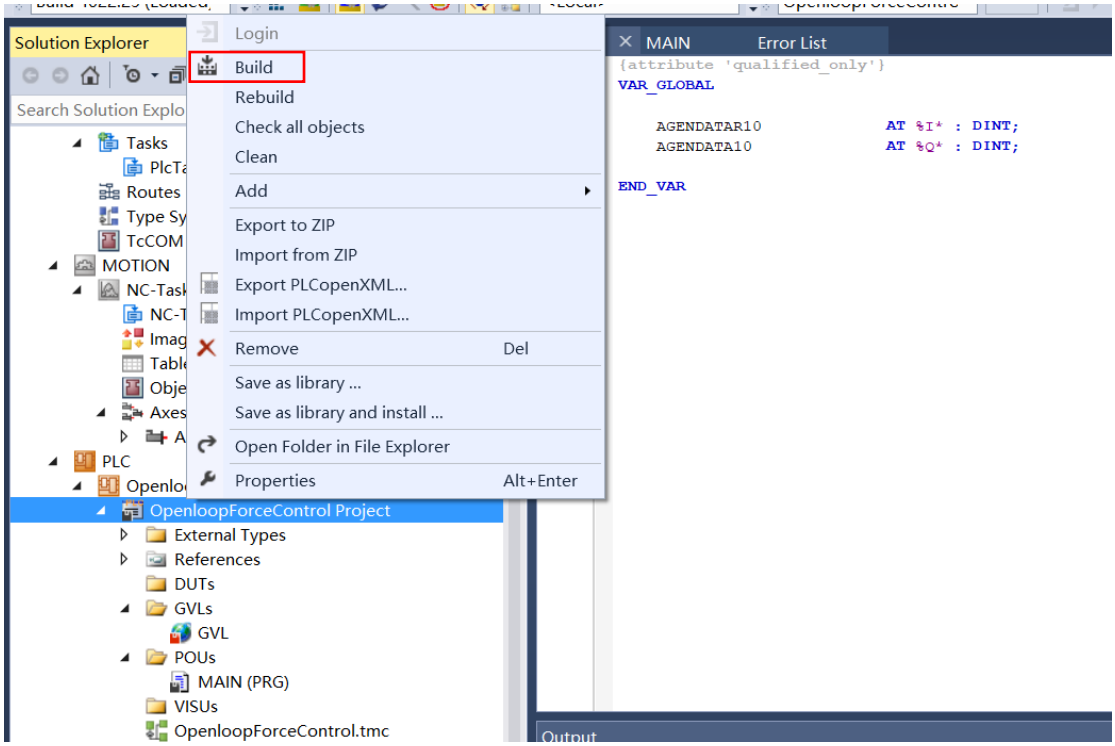
红色框下是定义的全局变量的名字，可以按照需求随意编写。

绿色框里面的内容根据数据是“Inport”或者是“Outport”来确定

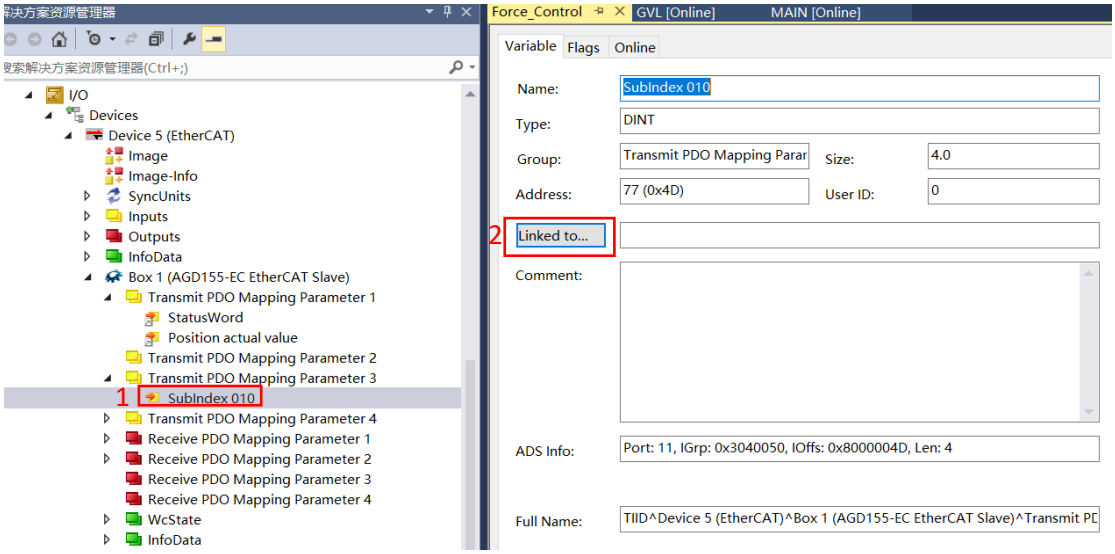
“Inport”为“%I*”，“Outport”为“%Q*”

蓝色框里面的内容根据数据类型填写

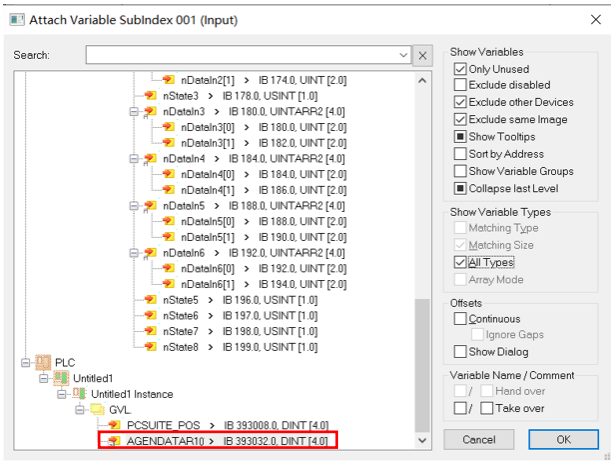
Name		Online	Type	Size	>Address	In/Out	User ...	Linked to
InputToggle	X		BIT	0.1	1524.3	Input	0	nState4, nState4
State		8	UINT	2.0	1548.0	Input	0	
AdsAddr		169.254.80.80.6....	AMSADDR	8.0	1550.0	Input	0	
Chn0		0	USINT	1.0	1558.0	Input	0	
DcOutputShift	X	87260	DINT	4.0	1559.0	Input	0	nDcOutputTime . In . I...
DcInputShift	X	412740	DINT	4.0	1563.0	Input	0	nDcInputTime . In . Inp...
ControlWord	X	31	UINT	2.0	71.0	Output	0	nCtrl1, nCtrl2
Target Position	X	0	DINT	4.0	73.0	Output	0	nDataOut1 . Out . Out...
Touch Probe F...	X		UINT	2.0	77.0	Output	0	nCtrl5, nCtrl6
SubIndex 001	X		DINT	4.0	79.0	Output	0	GVL.AGENDATA1 . Plc...
SubIndex 002	X		DINT	4.0	83.0	Output	0	GVL.AGENDATA2 . Plc...

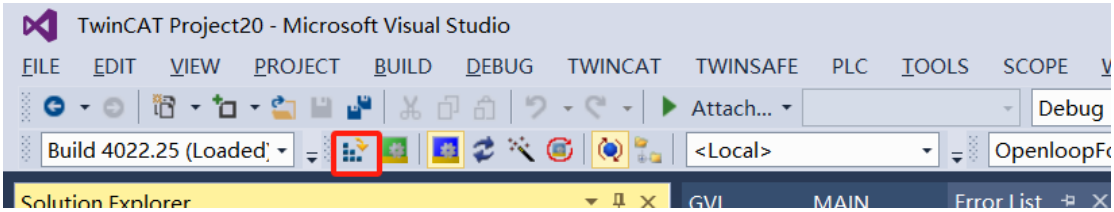


定义好全局变量后，对 PLC 进行编译。



找到刚才添加的 PDO，点击 link to 按钮。打开关联界面，选择好对应的全局变量点击“OK”。





完成相关配置后点击 Activate configuration 按钮保存相关配置。

之后在 PLC 程序中配置和读取控制器 AGendata 的数据了。

在程序中的使用办法是“GVL.xxxxx（定义的变量名称）”。

如图：

```
GVL.AGENDATA10:=1;
GVL.AGENDATAR10
```

2.1.2 PLC 与 IDE 程序流程

PLC 程序中负责整个力控流程的使能，报错，复位，运动部分。

IDE 程序负责力控流程中的切换力控模式，运动模式切换，回退部分。

PLC 程序给电机使能之后发送运动指令使电机往按压物体运动，当电机按压到物体，满足切换力控条件时，从站会把位置模式切换为电流模式。（此时主站发送运动指令只有 Target Position 会变化，但电机不会按照发送的运动指令运动，这就导致了实际位置与指令位置有了偏差。）此时我们在从站执行力控期间，主站发送运动指令，让 Target Position 提前到达电机做完力控的回退位置。在从站在执行力控期间需要将 Motionmode 转换为 1（PTP）后再执行回退运动，在回退到位后切换 Motionmode 为 19（CSP），在确认主站运动到位（Actual Position 和 Target Position 相等后），IDE 程序发送 Begin 再执行后续运动。

2.1.3PLC 与 IDE 程序内容

定义程序变量，数据类型以及初始值等

```
    //Write DriveOperationMode Function=============================================================
    fbWriteDriveOperationMode_MOTOR : MC_WriteDriveOperationMode;
    bWriteDriveOperationMode_Execute_MOTOR: BOOL;
    iWriteDriveOperationMode_MOTOR: E_DriveOperationMode;
    bWriteDriveOperationMode_Done_MOTOR: BOOL;
//MoveAbsolute Function=========================================================================
    fbMoveAbsolute_MOTOR : MC_MoveAbsolute;
    bMoveAbsolute_Execute_MOTOR: BOOL := FALSE;
    bMoveAbsolute_Done_MOTOR: BOOL;
    fTargetPosition_MOTOR: LREAL := 100;
    fTargetVelocity_MOTOR : LREAL := 30;
    fTargetAcceleration_MOTOR : LREAL := 300;
```

将定义好的变量和功能块参数关联起来

```
//Update status
MOTOR.ReadStatus();

//Reset Function
fbReset_MOTOR(Axis := MOTOR, Execute:= bReset_Execute_MOTOR, Done=> bReset_Done_MOTOR);

//Halt Function
fbHalt_MOTOR(Axis := MOTOR, Execute := bHalt_MOTOR_Execute, Done => bHalt_MOTOR_Done ,Deceleration := bHalt_MOTOR_Deceleration);

//Power Function
fbPower_MOTOR(Axis:= MOTOR, Enable:= bPower_Enable_MOTOR, Enable_Positive:= bPower_Enable_MOTOR, Enable_Negative:= bPower_Enable_MOTOR, Override:=

//WriteDriveOperationMode Function
fbWriteDriveOperationMode_MOTOR(Axis:= MOTOR, Execute:= bWriteDriveOperationMode_Execute_MOTOR, DriveOperationMode:=iWriteDriveOperationMode_MOTOR

//MoveAbsolute Function
fbMoveAbsolute_MOTOR(Axis:= MOTOR, Execute:= bMoveAbsolute_Execute_MOTOR, Position:= fTargetPosition_MOTOR, Velocity:= fTargetVelocity_MOTOR, Acce
```

整体的 PLC 程序是通过许多个 CASE 组成，当完成当前 CASE 时，即跳转到下一个 CASE。

```
CASE MOVING OF
    MOTOR_RESET:
        bPower_Enable_MOTOR  := FALSE;
        bReset_Execute_MOTOR := TRUE;
        //MOVING := MOTOR_ENABLE;
    MOTOR_ENABLE:
        bPower_Enable_MOTOR  := TRUE;
        bReset_Execute_MOTOR := FALSE;
        MOVING := MOTOR_ENABLEWAIT;
    MOTOR_ENABLEWAIT:
    IF bPower_Status_MOTOR THEN
        GVL.AGENDATA1 := 0;
        MOVING := OFF_AGENDATA1_WAIT;
    END_IF

    OFF_AGENDATA1_WAIT:
    IF GVL.AGENDATAR1 = 0 THEN
        MOVING := SEND_FORCECONTROLSIGNAL;
    END_IF

    SEND_FORCECONTROLSIGNAL:

    GVL.AGENDATA1 := 1;
    MOVING := SEND_FORCECONTROLSIGNAL_WAIT;

    SEND_FORCECONTROLSIGNAL_WAIT:
    IF GVL.AGENDATAR1 = 1 THEN
        MOVING := MOTOR_MOVING;
    ELSE
        MOVING :=SEND_FORCECONTROLSIGNAL_WAIT;
    END_IF

    MOTOR_MOVING:
        fTargetPosition_MOTOR := 100;//fTargetPosition_MOTOR ; // 100
        fTargetVelocity_MOTOR := fTargetVelocity_MOTOR ;
        fTargetAcceleration_MOTOR := 10*fTargetVelocity_MOTOR ;
        bMoveAbsolute_Execute_MOTOR := TRUE;
        MOVING := MOVING_WAITING;
```

PLC 完成使能后，在开始运动之前给发送 AGendata[1]=1,在 IDE 中开启力控程序。

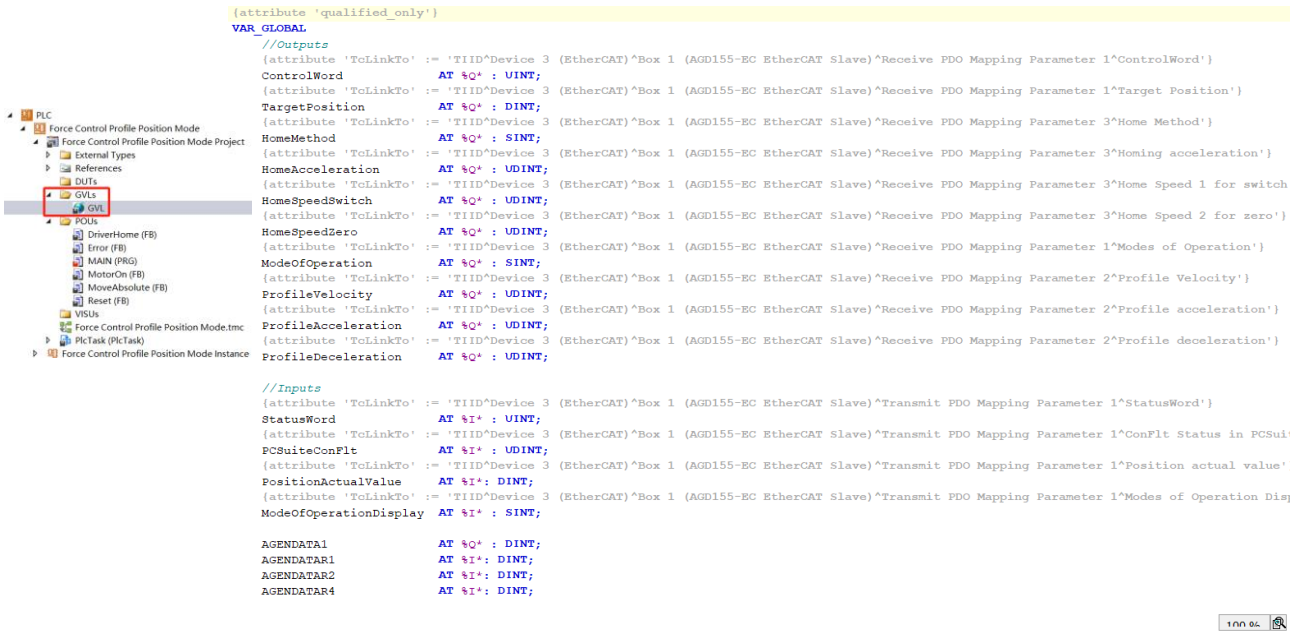
```
while (1)
    if (AGenData[1]==1)
        ACurrPosThDir=1
        ACurrPosTh=0
        ACurrCurrTh=500
        ACurrCurrThDir=0

        ACurrCmdSrc=1
        ACurrCmdSlope[1]=10000
        ACurrCmdHTime[1]=2000
        ACurrCmdVal[1]=800
        ACurrCmdSlope[2]=10000
        ACurrCmdHTime[2]=2000
        ACurrCmdVal[2]=1200
        while (AOperationMode!=1)
        end
        AGenData[4] = 1 // start force control
        AGenData[5] = 1
        AGenData[1] = 0
end
```

3 PP 模式下力控流程操作步骤

PP 模式使用 Controlword 和 Statusword 中的某些位用于模式特定的目的。PP 模式下的开环力控需要手动写对应的功能块去修改从站的 Controlword 和 Statusword 来实现对应的功能。在写对应功能块之前需要进行前文关联 AGendata 一样，定义全局变量，将全局变量和需要的 PDO 关联。由于 PP 模式始终是在 Motionmode 为 1 下运动，而且不存在 CSP 模式下 Actual Position 和 Target Position 不同的情况下，所以流程会比 CSP 模式下简单。PLC 程序发送使能指令后发送 AGendata 使 IDE 对一系列力控参数进行赋值，等待电机满足切换力控条件时切换为力控模式。

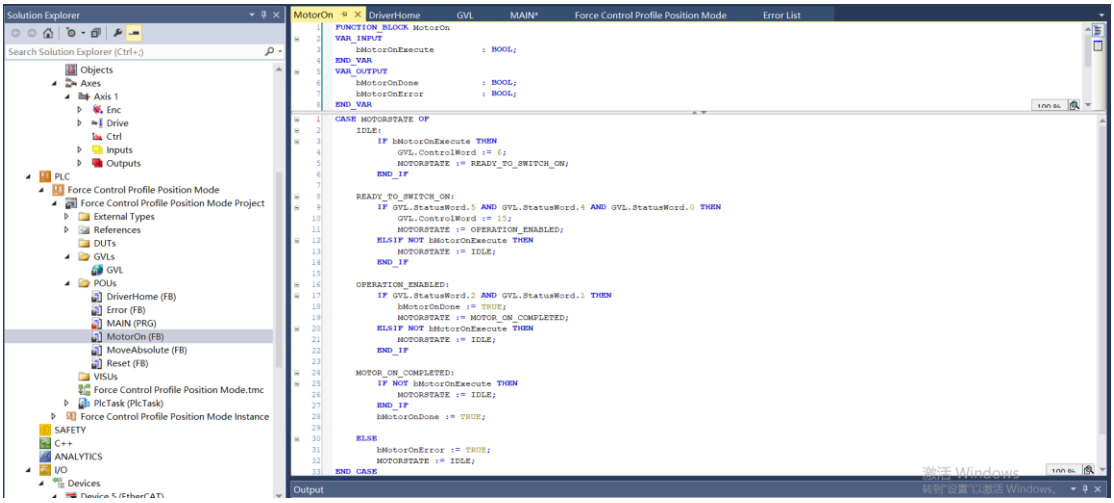
执行完力控程序后返回到回退位置后可以直接执行后续操作。

3.1 配置流程



需要先将需要的参数在 GVL 中定义好全局变量，编译好后将全局变量与对应的 PDO 关联起来。然后在 POUs 中定义对应功能块。定义好后在主程序中调用即可。

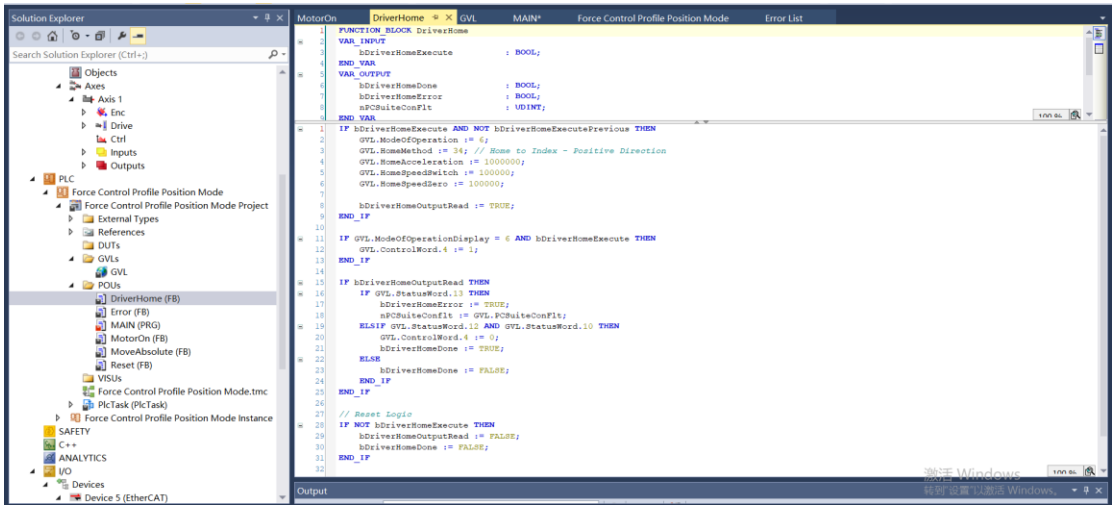
使能：



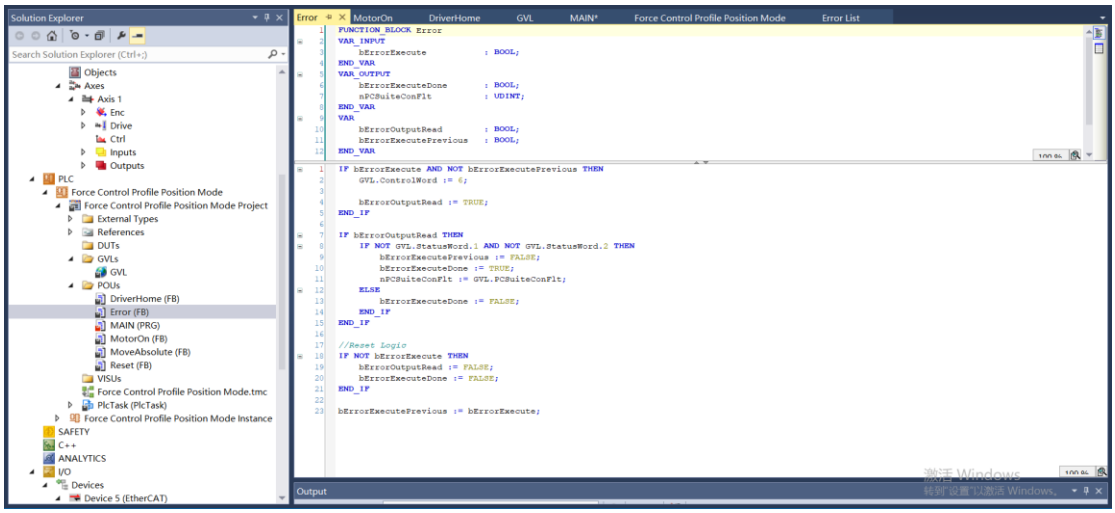
复位：


Solution Explorer
Search Solution Explorer (Ctrl+;)
Objects
Axes
Axis 1
Enc
Drive
Ctrl
Inputs
Outputs
PLC
Force Control Profile Position Mode
Force Control Profile Position Mode Project
External Types
References
DUTs
GVLs
GVL
POUs
DriverHome (FB)
Error (FB)
MAIN (PRG)
MotorOn (FB)
MoveAbsolute (FB)
Reset (FB)
VISUs
Force Control Profile Position Mode.tmc
PlcTask (PlcTask)
Reset
MoveAbsolute
Error
MotorOn
DriverHome
GVL
MAIN*
Force Control Profile Position Mode
Error List
FUNCTION_BLOCK Reset
VAR_INPUT
bResetExecute : BOOL;
END_VAR
VAR_OUTPUT
bResetExecuteDone : BOOL;
END_VAR
VAR
bResetExecutePrevious : BOOL;
bResetOutputRead : BOOL;
END_VAR
IF bResetExecute AND NOT bResetExecutePrevious THEN
GVL.ControlWord.7 := 1;
bResetOutputRead := TRUE;
END_IF
IF bResetOutputRead THEN
IF NOT GVL.StatusWord.3 THEN
GVL.ControlWord.7 := 0;
bResetExecuteDone := TRUE;
ELSE
bResetExecuteDone := FALSE;
END_IF
END_IF
// Reset Logic
IF NOT bResetExecute THEN
bResetOutputRead := FALSE;
bResetExecuteDone := FALSE;
END_IF
bResetExecutePrevious := bResetExecute;

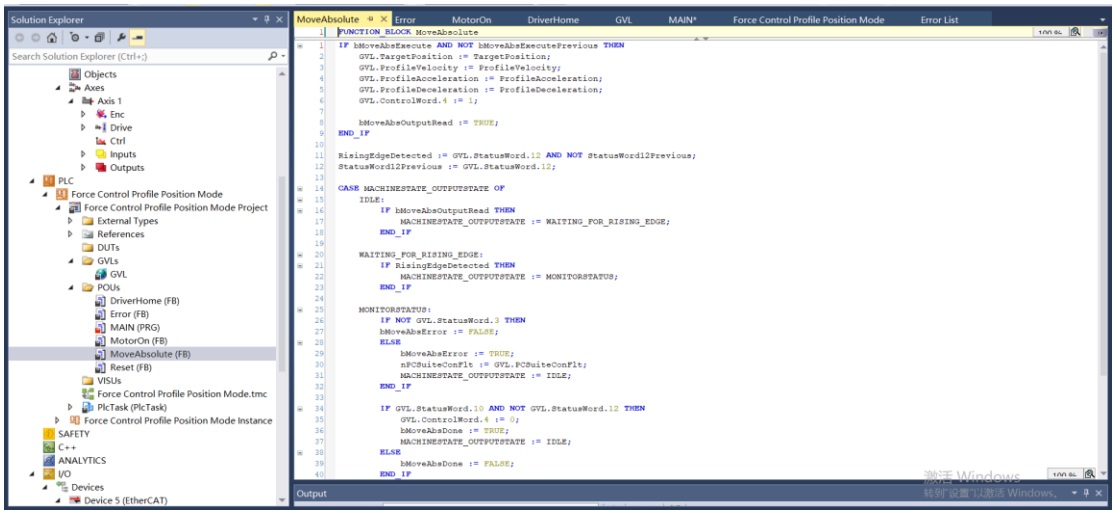
回零：


Solution Explorer
Search Solution Explorer (Ctrl+;)
Objects
Axes
Axis 1
Enc
Drive
Ctrl
Inputs
Outputs
PLC
Force Control Profile Position Mode
Force Control Profile Position Mode Project
External Types
References
DUTs
GVLs
GVL
POUs
DriverHome (FB)
Error (FB)
MAIN (PRG)
MotorOn (FB)
MoveAbsolute (FB)
Reset (FB)
VISUs
Force Control Profile Position Mode.tmc
PlcTask (PlcTask)
Force Control Profile Position Mode Instance
SAFETY
C++
ANALYTICS
I/O
Devices
Device 5 (EtherCAT)
MotorOn
DriverHome
GVL
MAIN*
Force Control Profile Position Mode
Error List
FUNCTION_BLOCK DriverHome
VAR_INPUT
bDriverHomeExecute : BOOL;
END_VAR
VAR_OUTPUT
bDriverHomeDone : BOOL;
bDriverHomeError : BOOL;
nPCSuiteConFlt : UDINT;
END_VAR
IF bDriverHomeExecute AND NOT bDriverHomeExecutePrevious THEN
GVL.ModeOfOperation := 6;
GVL.HomeMethod := 34; // Home to Index - Positive Direction
GVL.HomeAcceleration := 1000000;
GVL.HomeSpeedSwitch := 100000;
GVL.HomeSpeedZero := 100000;
bDriverHomeOutputRead := TRUE;
END_IF
IF GVL.ModeOfOperationDisplay = 6 AND bDriverHomeExecute THEN
GVL.ControlWord.4 := 1;
END_IF
IF bDriverHomeOutputRead THEN
IF GVL.StatusWord.13 THEN
bDriverHomeError := TRUE;
nPCSuiteConFlt := GVL.PCSuiteConFlt;
ELSIF GVL.StatusWord.12 AND GVL.StatusWord.10 THEN
GVL.ControlWord.4 := 0;
bDriverHomeDone := TRUE;
ELSE
bDriverHomeDone := FALSE;
END_IF
END_IF
// Reset Logic
IF NOT bDriverHomeExecute THEN
bDriverHomeOutputRead := FALSE;
bDriverHomeDone := FALSE;
END_IF
Output

报错：

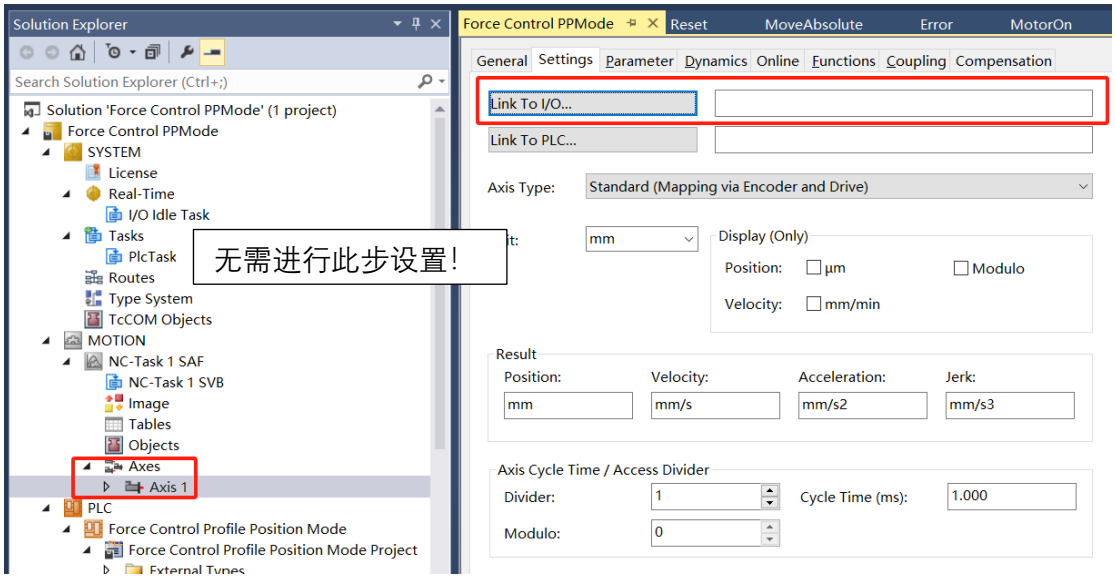

Solution Explorer
Search Solution Explorer (Ctrl+;)
Objects
Axes
Axis 1
Enc
Drive
Ctrl
Inputs
Outputs
PLC
Force Control Profile Position Mode
Force Control Profile Position Mode Project
External Types
References
DUTs
GVLs
GVL
POUs
DriverHome (FB)
Error (FB)
MAIN (PRG)
MotorOn (FB)
MoveAbsolute (FB)
Reset (FB)
VISUs
Force Control Profile Position Mode.tmc
PlcTask (PlcTask)
Force Control Profile Position Mode Instance
SAFETY
C++
ANALYTICS
I/O
Devices
Device 5 (EtherCAT)
Error
MotorOn
DriverHome
GVL
MAIN*
Force Control Profile Position Mode
Error List
FUNCTION_BLOCK Error
VAR_INPUT
bErrorExecute : BOOL;
END_VAR
VAR_OUTPUT
bErrorExecuteDone : BOOL;
nPCSuiteConFlt : UDINT;
END_VAR
VAR
bErrorOutputRead : BOOL;
bErrorExecutePrevious : BOOL;
END_VAR
IF bErrorExecute AND NOT bErrorExecutePrevious THEN
GVL.ControlWord := 6;
bErrorOutputRead := TRUE;
END_IF
IF bErrorOutputRead THEN
IF NOT GVL.StatusWord.1 AND NOT GVL.StatusWord.2 THEN
bErrorExecutePrevious := FALSE;
bErrorExecuteDone := TRUE;
nPCSuiteConFlt := GVL.PCSuiteConFlt;
ELSE
bErrorExecuteDone := FALSE;
END_IF
END_IF
//Reset Logic
IF NOT bErrorExecute THEN
bErrorOutputRead := FALSE;
bErrorExecuteDone := FALSE;
END_IF
bErrorExecutePrevious := bErrorExecute;
Output

绝对运动：



如果需要其它功能的功能块请查询 402 手册并自行编写。

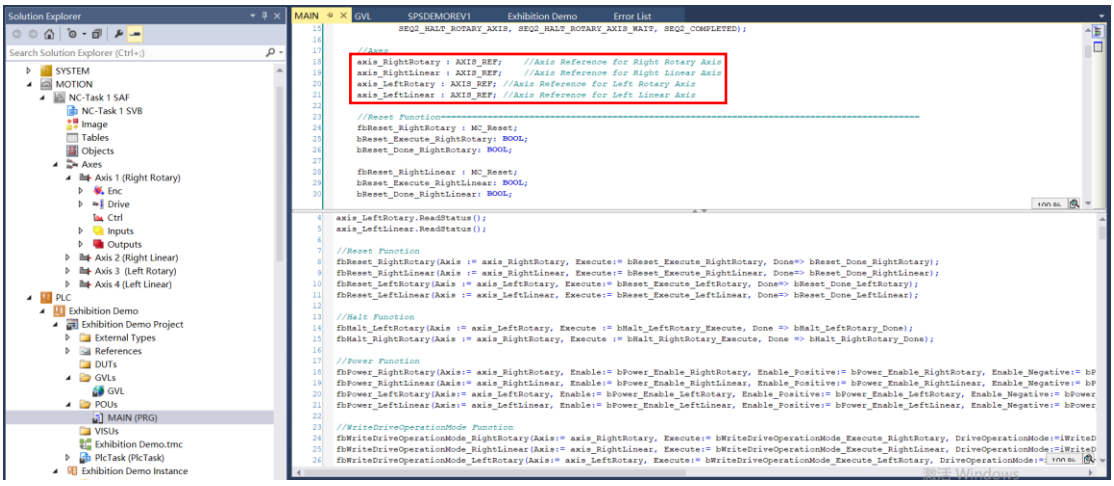
注意事项：使用 PP 模式时无需再进行 CSP 模式下关联 I/O 的设置，直接使用即可。



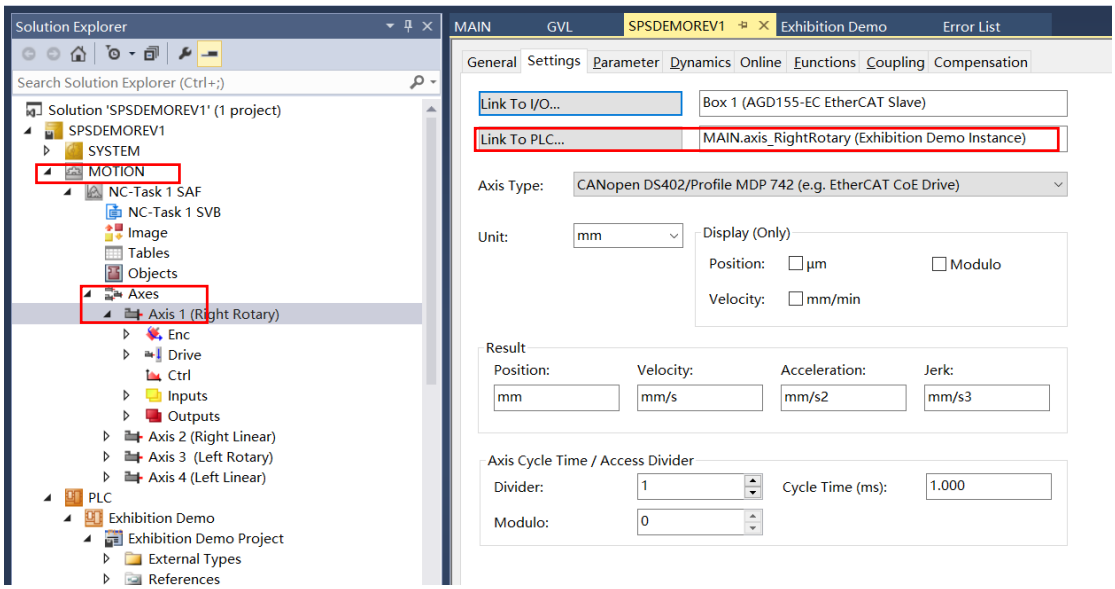
3.2 PLC 与 IDE 程序内容

4 配置多轴的方法

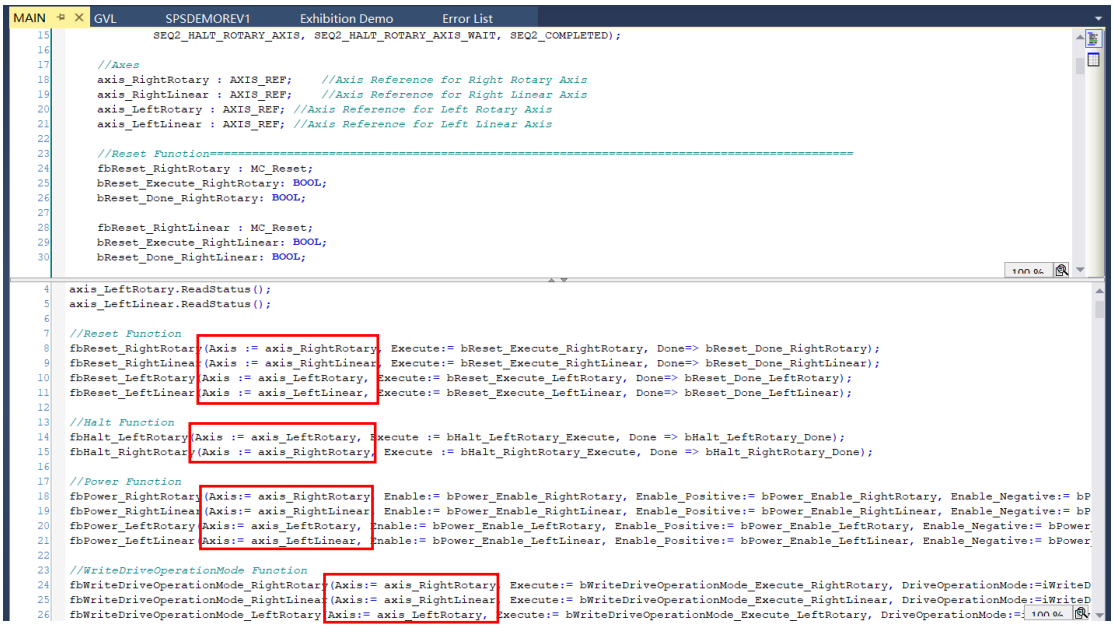
4.1 CSP 模式



在主程序中变量区位置使用 AXIS_REF 功能块，定义轴名称。



在 Motion→Axes→Setting 界面 Link to PLC 中将刚刚定义好的轴名称与对应轴关联起来。



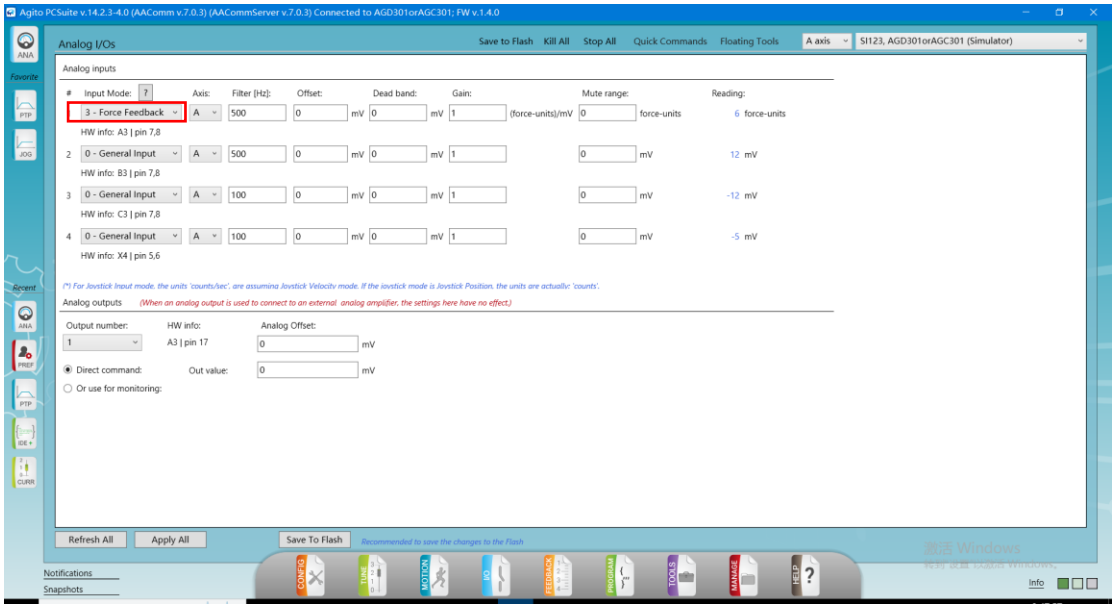
在主程序中将功能块中的 Axis 选项选择对应轴，在调用不同功能块时会给不同轴发指令。并在主程序中按照不同力控应用设计力控程序。

4.2 PP 模式

如果要在 PP 模式下实现多轴开环力控的话，需要将每个轴的需要的 PDO 先在 GVL 中定义全局变量并将各个轴的 PDO 与相应全局变量对应起来。然后每个轴重复单轴的力控流程操作（编写功能块→主程序中调用功能块配合 IDE 程序实现力控流程），根据实际应用实现不同的多轴力控。

5 配置闭环力控的方法

配置闭环力控的方法跟开环力控的方法类似，Twincat 端的配置相同，在 PCsuite 中进行闭环力控的相关配置，包括力传感器模拟量接入，模拟量类型修改，模拟量标定等（具体细节请查阅 Agito 力控手册）。此后将 PCsuite 中的 IDE 程序开环力控部分修改为闭环力控部分即可执行 Ethercat 通讯下的闭环力控。



```
            AForceAInTh=50//预设5g 进入力控 operationmode=4 需要条件时满足
            AForceCmdSrc=1 //力控命令来源
            AForceCmdSlope[1]=4000//K
            AForceCmdHTime[1]=2000//停留时间 ms
            AForceCmdVal[1]=200//设置按压力值
            AForceCmdHTime[2]=0//压合结束 Index根据段数+1
//          ACurrPosThDir=1
//          ACurrPosTh=0
//          ACurrCurrTh=500
//          ACurrCurrThDir=0
//
//          ACurrCmdSrc=1
//          ACurrCmdSlope[1]=10000
//          ACurrCmdHTime[1]=2000
//          ACurrCmdVal[1]=800
//          ACurrCmdSlope[2]=10000
//          ACurrCmdHTime[2]=2000
//          ACurrCmdVal[2]=1200
```

闭环力控 IDE 程序部分

开环力控 IDE 程序部分