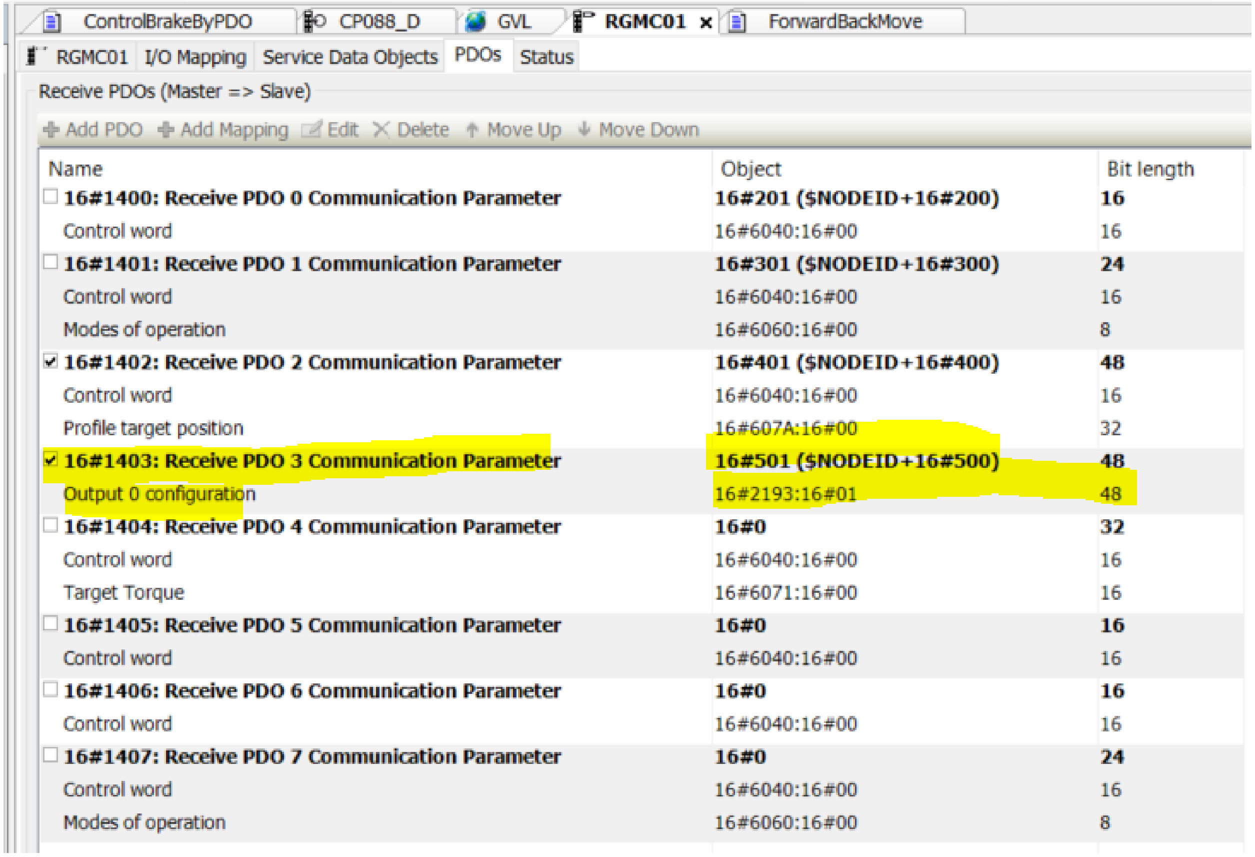
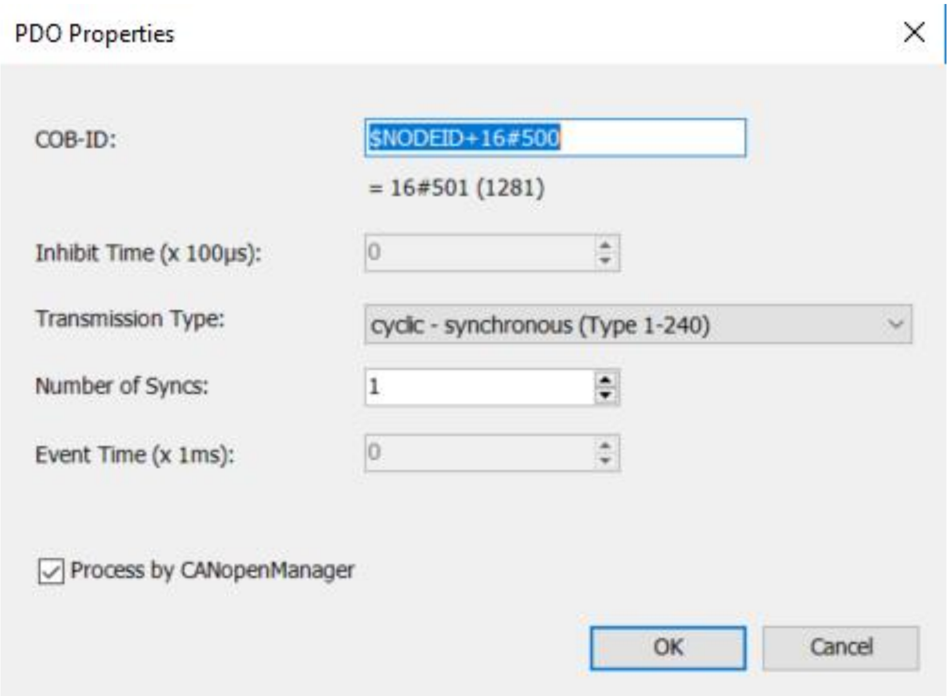
Table of Contents

1. 在 PDO 配置界面添加参数



Receive PDOs (Master => Slave)

Name	Object	Bit length
□ **16#1400: Receive PDO 0 Communication Parameter**	**16#201 ($NODEID+16#200)**	**16**
Control word	16#6040:16#00	16
□ **16#1401: Receive PDO 1 Communication Parameter**	**16#301 ($NODEID+16#300)**	**24**
Control word	16#6040:16#00	16
Modes of operation	16#6060:16#00	8
☑ **16#1402: Receive PDO 2 Communication Parameter**	**16#401 ($NODEID+16#400)**	**48**
Control word	16#6040:16#00	16
Profile target position	16#607A:16#00	32
☑ **16#1403: Receive PDO 3 Communication Parameter**	**16#501 ($NODEID+16#500)**	**48**
Output 0 configuration	16#2193:16#01	48
□ **16#1404: Receive PDO 4 Communication Parameter**	**16#0**	**32**
Control word	16#6040:16#00	16
Target Torque	16#6071:16#00	16
□ **16#1405: Receive PDO 5 Communication Parameter**	**16#0**	**16**
Control word	16#6040:16#00	16
□ **16#1406: Receive PDO 6 Communication Parameter**	**16#0**	**16**
Control word	16#6040:16#00	16
□ **16#1407: Receive PDO 7 Communication Parameter**	**16#0**	**24**
Control word	16#6040:16#00	16
Modes of operation	16#6060:16#00	8



PDO Properties

COB-ID: $NODEID+16#500

= 16#501 (1281)

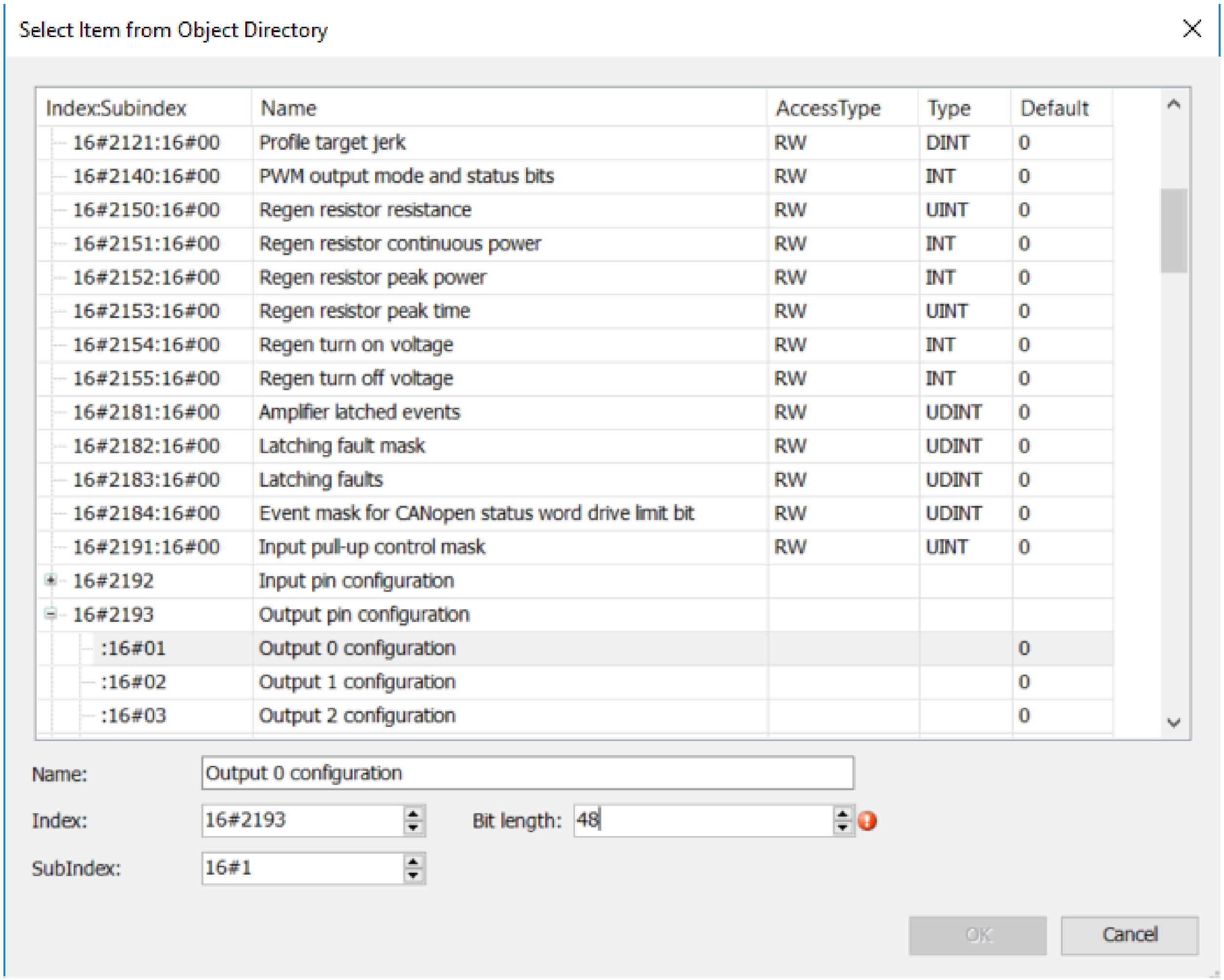
Inhibit Time (x 100µs): 0

Transmission Type: cyclic - synchronous (Type 1-240)

Number of Syncs: 1

Event Time (x 1ms): 0

☑ Process by CANopenManager



Select Item from Object Directory

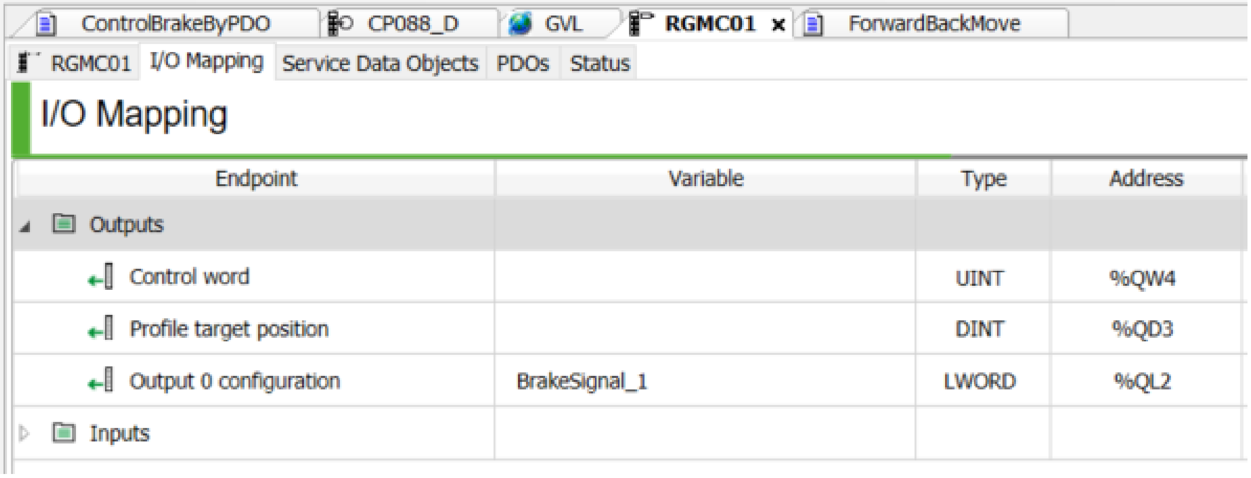
Index:Subindex	Name	AccessType	Type	Default
16#2121:16#00	Profile target jerk	RW	DINT	0
16#2140:16#00	PWM output mode and status bits	RW	INT	0
16#2150:16#00	Regen resistor resistance	RW	UINT	0
16#2151:16#00	Regen resistor continuous power	RW	INT	0
16#2152:16#00	Regen resistor peak power	RW	INT	0
16#2153:16#00	Regen resistor peak time	RW	UINT	0
16#2154:16#00	Regen turn on voltage	RW	INT	0
16#2155:16#00	Regen turn off voltage	RW	INT	0
16#2181:16#00	Amplifier latched events	RW	UDINT	0
16#2182:16#00	Latching fault mask	RW	UDINT	0
16#2183:16#00	Latching faults	RW	UDINT	0
16#2184:16#00	Event mask for CANopen status word drive limit bit	RW	UDINT	0
16#2191:16#00	Input pull-up control mask	RW	UINT	0
⊞ 16#2192	Input pin configuration			
⊟ 16#2193	Output pin configuration			
:16#01	Output 0 configuration			0
:16#02	Output 1 configuration			0
:16#03	Output 2 configuration			0

Name: Output 0 configuration

Index: 16#2193 Bit length: 48

SubIndex: 16#1

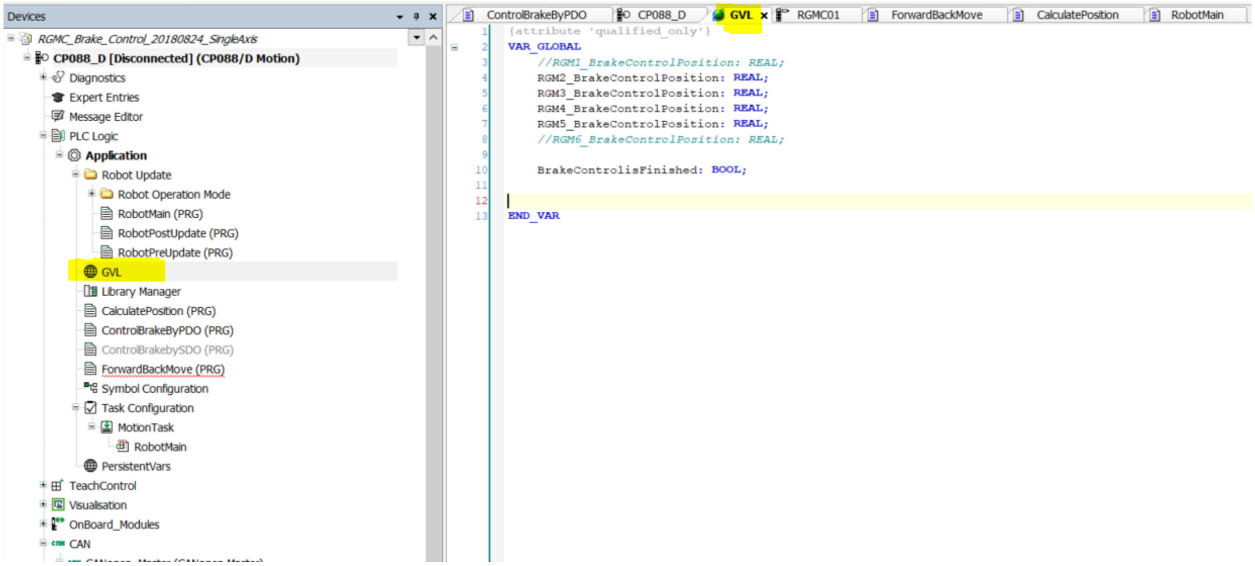
2. POD 中配置的参数设置表变量名



I/O Mapping

Endpoint	Variable	Type	Address
Outputs			
Control word		UINT	%QW4
Profile target position		DINT	%QD3
Output 0 configuration	BrakeSignal_1	LWORD	%QL2
Inputs			

3. 定义若干全局变量



```
{attribute 'qualified_only'}
VAR_GLOBAL
    //RGM1_BrakeControlPosition: REAL;
    RGM2_BrakeControlPosition: REAL;
    RGM3_BrakeControlPosition: REAL;
    RGM4_BrakeControlPosition: REAL;
    RGM5_BrakeControlPosition: REAL;
    //RGM6_BrakeControlPosition: REAL;

    BrakeControlisFinished: BOOL;


END_VAR
```

4. 编写程序 1

变量定义

```
PROGRAM CalculatePosition
VAR
        EnableRisingEdge: R_TRIG;
        BrakeOffset: LREAL := 0.30;//range from 0.10 to 0.80
        EnableCount: INT;
        RobotIsEnabled: BOOL;
END_VAR
```

程序主体部分

```
//BrakeOffset: 0.1 to 0.8
//detect robot state
IF GeneralRobot.PowerStatus THEN
```

```
    RobotIsEnabled := TRUE;
ELSE
    RobotIsEnabled := FALSE;
END_IF;
//rising edge of enable signal
EnableRisingEdge( CLK := RobotIsEnabled );
//calculate position for releasing brake
IF EnableRisingEdge.Q AND GeneralRobot.IsHomed THEN
    IF Axis2.ActualPosition > 0 THEN
        GVL.RGM2_BrakeControlPosition := LREAL_TO_REAL(Axis2.ActualPosition - BrakeOffset);
    ELSE
        GVL.RGM2_BrakeControlPosition := LREAL_TO_REAL(Axis2.ActualPosition + BrakeOffset);
    END_IF
    IF (Axis3.ActualPosition- Axis2.ActualPosition)> 0 THEN
        GVL.RGM3_BrakeControlPosition := LREAL_TO_REAL(Axis3.ActualPosition - BrakeOffset);
    ELSE
        GVL.RGM3_BrakeControlPosition := LREAL_TO_REAL(Axis3.ActualPosition + BrakeOffset);
    END_IF
    IF (Axis4.ActualPosition - (Axis3.ActualPosition- Axis2.ActualPosition)) > 0 THEN
        GVL.RGM4_BrakeControlPosition := LREAL_TO_REAL(Axis4.ActualPosition - BrakeOffset);
    ELSE
        GVL.RGM4_BrakeControlPosition := LREAL_TO_REAL(Axis4.ActualPosition + BrakeOffset);
    END_IF
    IF (Axis3.ActualPosition- Axis2.ActualPosition)> 0 THEN
        IF (Axis5.ActualPosition - (-90)) > 0 THEN
            GVL.RGM5_BrakeControlPosition := LREAL_TO_REAL(Axis5.ActualPosition - BrakeOffset);
        ELSE
            GVL.RGM5_BrakeControlPosition := LREAL_TO_REAL(Axis5.ActualPosition + BrakeOffset);
        END_IF
    ELSE
        IF (Axis5.ActualPosition - 90) > 0 THEN
            GVL.RGM5_BrakeControlPosition := LREAL_TO_REAL(Axis5.ActualPosition - BrakeOffset);
        ELSE
            GVL.RGM5_BrakeControlPosition := LREAL_TO_REAL(Axis5.ActualPosition + BrakeOffset);
        END_IF
    END_IF
    EnableCount := EnableCount + 1;
END_IF
```

5. 编写程序 2

变量定义部分

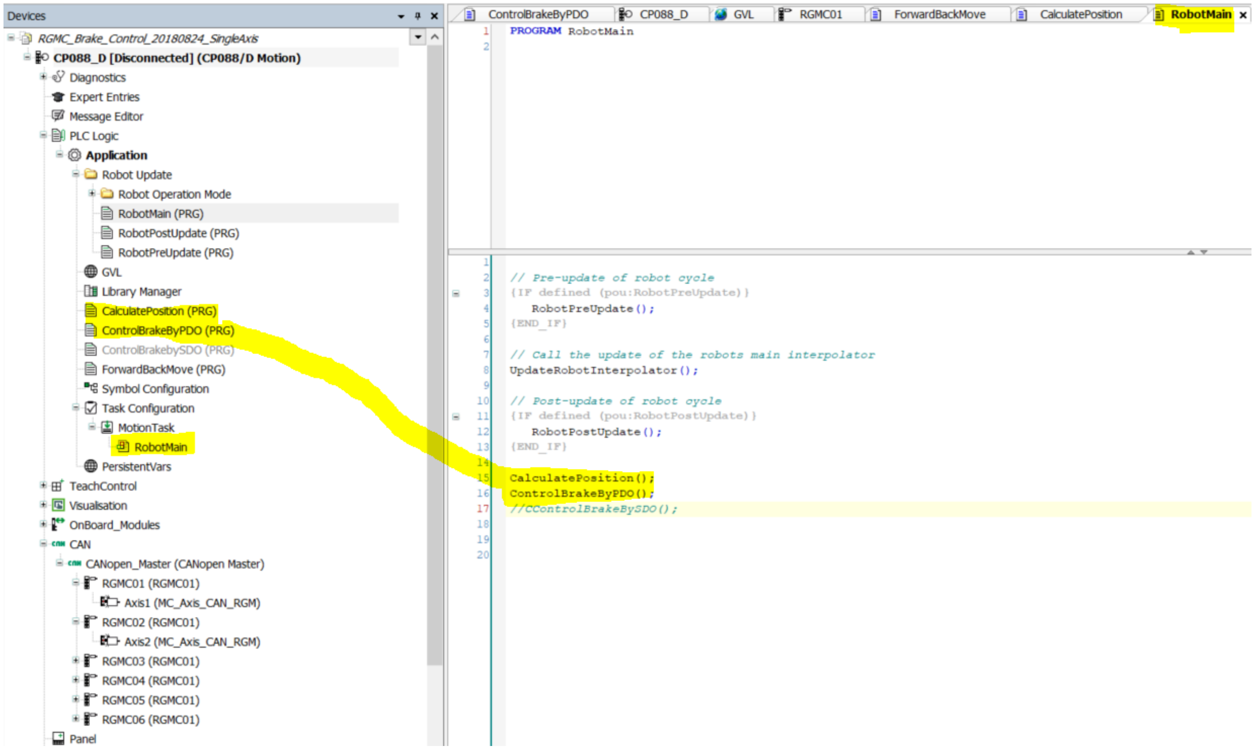
```
PROGRAM ControlBrakeByPDO
VAR
    myPowerOff: BOOL;
    myPowerOn: BOOL;
    PowerOffRisingEdge: R_TRIG;
    PowerOnRisingEdge: R_TRIG;
    DelayTime1: TON;
    DelayTime2: TON;
```

```
        myVar2: BOOL;
        myVar1: BOOL;
        mycount1: INT;
        mycount2: INT;
        PowerIsOff: BOOL;
        PowerIsOn: BOOL;
END_VAR
```

程序主体部分

```
IF GeneralRobot.PowerStatus = FALSE THEN
        myPowerOff := TRUE;
        myPowerOn := FALSE;
ELSE
        myPowerOff := FALSE;
        myPowerOn := TRUE;
END_IF

PowerOffRisingEdge(CLK:= myPowerOff);
PowerOnRisingEdge(CLK:= myPowerOn);
IF PowerOffRisingEdge.Q THEN
        PowerIsOff := TRUE;
        myVar2 := TRUE;
END_IF
IF PowerOnRisingEdge.Q THEN
        PowerIsOn := TRUE;
        myVar1 := TRUE;
END_IF

DelayTime1(IN:= myVar1, PT:= T#1S, Q=> , ET=> );
DelayTime2(IN:= myVar2, PT:= T#1S, Q=> , ET=> );

IF PowerIsOn AND DelayTime1.Q AND GVL.BrakeControlisFinished THEN
        mycount1 := mycount1 + 1;
        BrakeSignal_1 := 16#40000100;
        BrakeSignal_2 := 16#40000100;
        BrakeSignal_3 := 16#40000100;
        BrakeSignal_4 := 16#40000100;
        BrakeSignal_5 := 16#40000100;
        BrakeSignal_6 := 16#40000100;
        PowerIsOn := FALSE;
        myVar1 := FALSE;
        GVL.BrakeControlisFinished := FALSE;
END_IF;

IF PowerIsOff AND DelayTime2.Q THEN
        mycount2 := mycount2 + 1;
        //BrakeSignal_1 := 16#0;
        BrakeSignal_2 := 16#0;
        BrakeSignal_3 := 16#0;
        BrakeSignal_4 := 16#0;
        BrakeSignal_5 := 16#0;
        //BrakeSignal_6 := 16#0;
        PowerIsOff := FALSE;
        myVar2 := FALSE;
END_IF
```

6. 在主程序中调用两个子程序

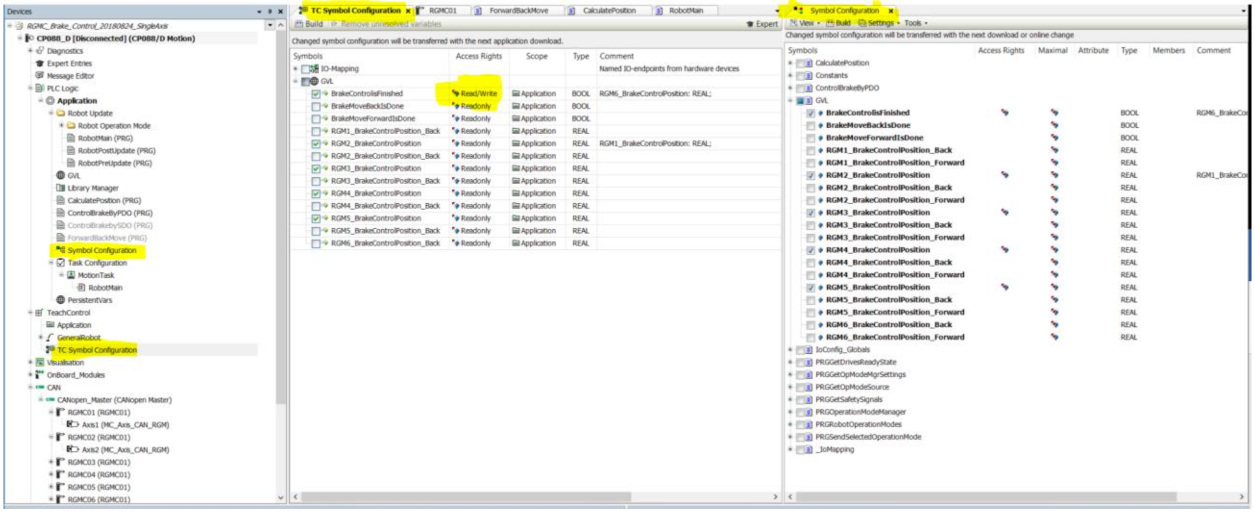


```
PROGRAM RobotMain
```

```

// Pre-update of robot cycle
{IF defined (pou:RobotPreUpdate)}
    RobotPreUpdate();
{END_IF}

// Call the update of the robots main interpolator
UpdateRobotInterpolator();

// Post-update of robot cycle
{IF defined (pou:RobotPostUpdate)}
    RobotPostUpdate();
{END_IF}

CalculatePosition();
ControlBrakeByPDO();
//CControlBrakeBySDO();
```

7. 打开 TC Symbol 和 Symbol Config， 勾选变量

注意更改下面的那个 BOOL 变量为可读可写



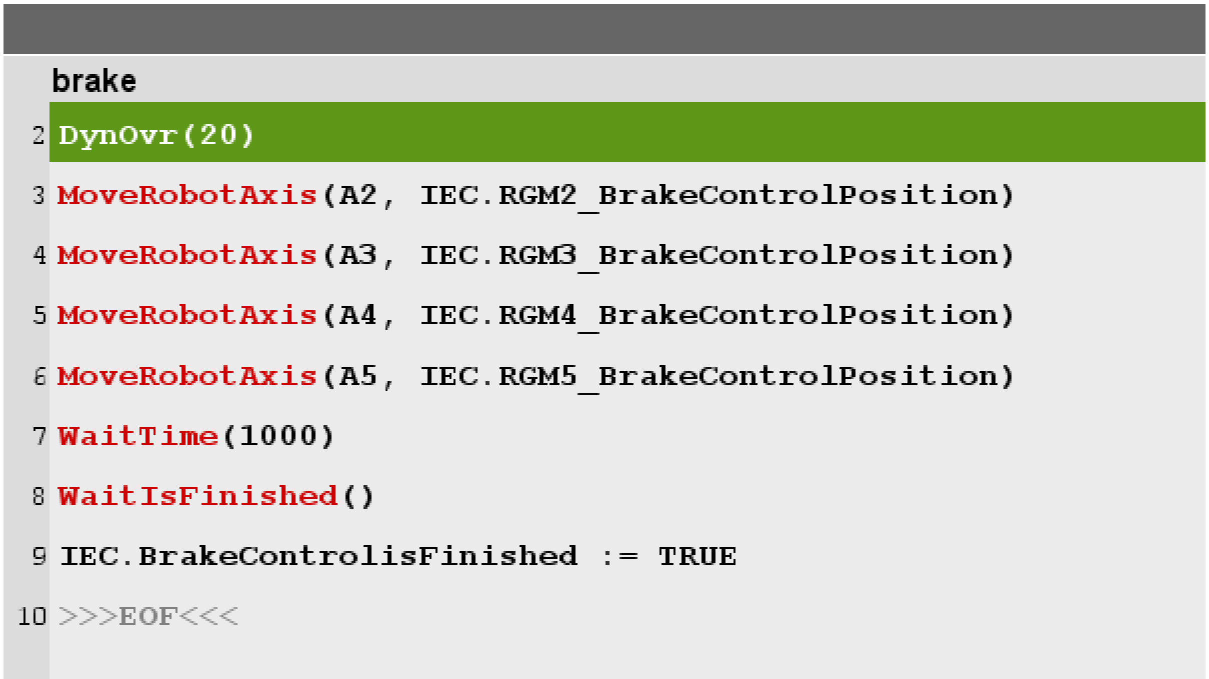
Changed symbol configuration will be transferred with the next application download.

Symbols	Access Rights	Scope	Type	Comment
□ IO-Mapping				Named IO-endpoints from hardware devices
GVL				
☑ BrakeControlisFinished	Read/Write	Application	BOOL	RGM6_BrakeControlPosition: REAL;
□ BrakeMoveBackIsDone	Readonly	Application	BOOL	
□ BrakeMoveForwardIsDone	Readonly	Application	BOOL	
□ RGM1_BrakeControlPosition_Back	Readonly	Application	REAL	
☑ RGM2_BrakeControlPosition	Readonly	Application	REAL	RGM1_BrakeControlPosition: REAL;
□ RGM2_BrakeControlPosition_Back	Readonly	Application	REAL	
☑ RGM3_BrakeControlPosition	Readonly	Application	REAL	
□ RGM3_BrakeControlPosition_Back	Readonly	Application	REAL	
☑ RGM4_BrakeControlPosition	Readonly	Application	REAL	
□ RGM4_BrakeControlPosition_Back	Readonly	Application	REAL	
☑ RGM5_BrakeControlPosition	Readonly	Application	REAL	
□ RGM5_BrakeControlPosition_Back	Readonly	Application	REAL	
□ RGM6_BrakeControlPosition_Back	Readonly	Application	REAL	

Changed symbol configuration will be transferred with the next download or online change

Symbols	Access Rights	Maximal	Attribute	Type	Members	Comment
□ CalculatePosition						
□ Constants						
□ ControlBrakeByPDO						
GVL						
☑ BrakeControlisFinished	✓	✓		BOOL		RGM6_BrakeCo
□ BrakeMoveBackIsDone		✓		BOOL		
□ BrakeMoveForwardIsDone		✓		BOOL		
□ RGM1_BrakeControlPosition_Back		✓		REAL		
□ RGM1_BrakeControlPosition_Forward		✓		REAL		
☑ RGM2_BrakeControlPosition	✓	✓		REAL		RGM1_BrakeCo
□ RGM2_BrakeControlPosition_Back		✓		REAL		
□ RGM2_BrakeControlPosition_Forward		✓		REAL		
☑ RGM3_BrakeControlPosition	✓	✓		REAL		
□ RGM3_BrakeControlPosition_Back		✓		REAL		
□ RGM3_BrakeControlPosition_Forward		✓		REAL		
☑ RGM4_BrakeControlPosition	✓	✓		REAL		
□ RGM4_BrakeControlPosition_Back		✓		REAL		
□ RGM4_BrakeControlPosition_Forward		✓		REAL		
☑ RGM5_BrakeControlPosition	✓	✓		REAL		
□ RGM5_BrakeControlPosition_Back		✓		REAL		
□ RGM5_BrakeControlPosition_Forward		✓		REAL		
□ RGM6_BrakeControlPosition_Back		✓		REAL		
□ RGM6_BrakeControlPosition_Forward		✓		REAL		
□ IoConfig_Globals						
□ PRGGetDrivesReadyState						
□ PRGGetOpModeMgrSettings						
□ PRGGetOpModeSource						
□ PRGGetSafetySignals						
□ PRGOperationModeManager						
□ PRGRobotOperationModes						
□ PRGSendSelectedOperationMode						
□ _IoMapping						

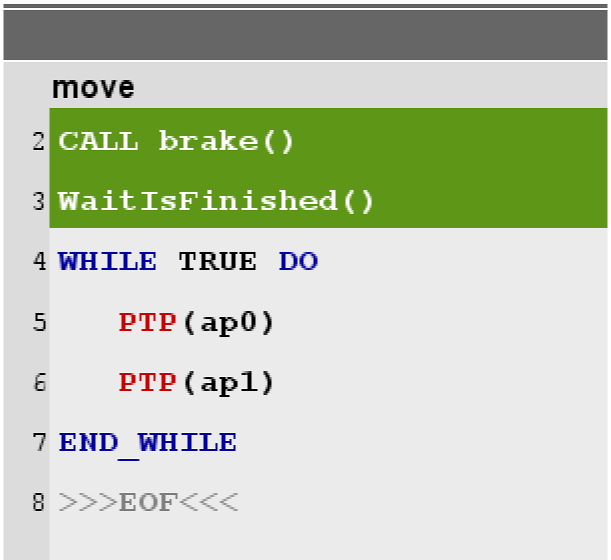
8. 在示教器中编写程序



brake

```
DynOvr(20)
MoveRobotAxis(A2, IEC.RGM2_BrakeControlPosition)
MoveRobotAxis(A3, IEC.RGM3_BrakeControlPosition)
MoveRobotAxis(A4, IEC.RGM4_BrakeControlPosition)
MoveRobotAxis(A5, IEC.RGM5_BrakeControlPosition)
WaitTime(1000)
WaitIsFinished()
IEC.BrakeControlisFinished := TRUE
>>>EOF<<<
```

9. 在示教器中调用程序



move

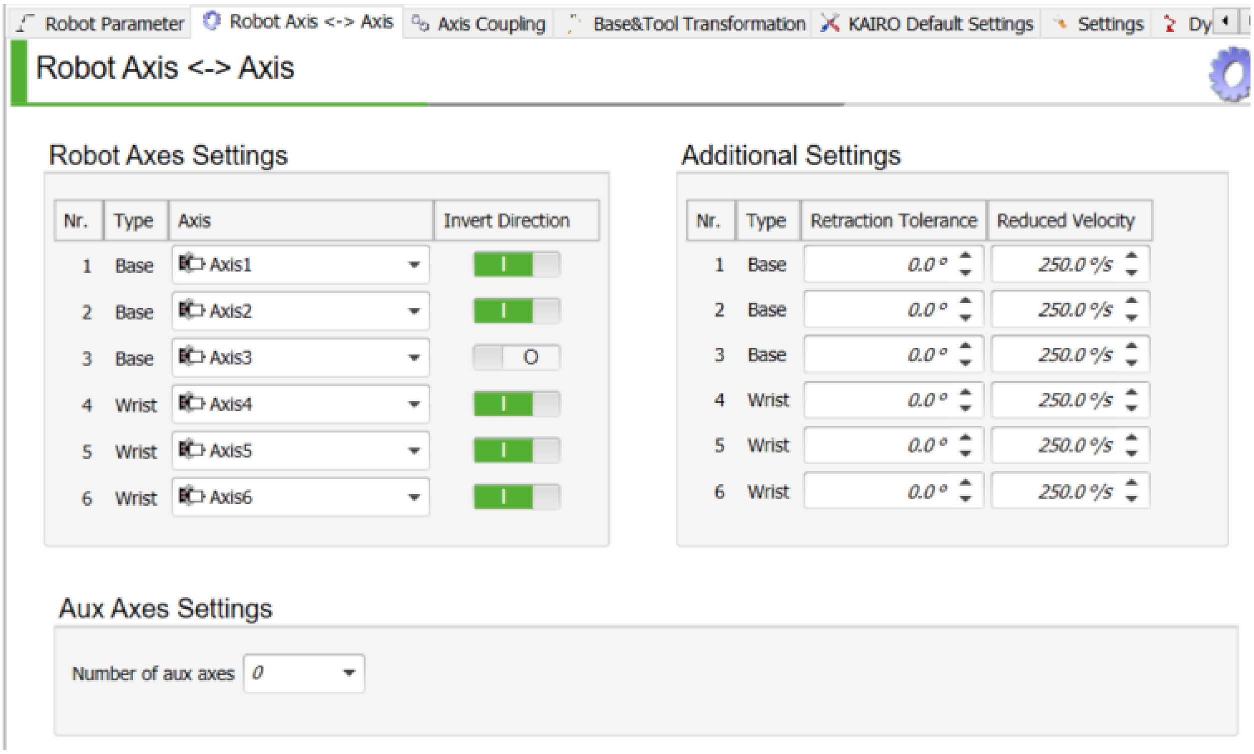
```
CALL brake()
WaitIsFinished()
WHILE TRUE DO
    PTP(ap0)
    PTP(ap1)
END_WHILE
>>>EOF<<<
```

10. 若干提示

a. 机器人必须处于回零模式

b. 断电后，最好不要手动移动机器人的状态

c. 机器人轴的方向要和本程序中的一致



Robot Axis <-> Axis

Robot Axes Settings

Nr.	Type	Axis	Invert Direction
1	Base	Axis1	I
2	Base	Axis2	I
3	Base	Axis3	O
4	Wrist	Axis4	I
5	Wrist	Axis5	I
6	Wrist	Axis6	I

Additional Settings

Nr.	Type	Retraction Tolerance	Reduced Velocity
1	Base	0.0 °	250.0 °/s
2	Base	0.0 °	250.0 °/s
3	Base	0.0 °	250.0 °/s
4	Wrist	0.0 °	250.0 °/s
5	Wrist	0.0 °	250.0 °/s
6	Wrist	0.0 °	250.0 °/s

Aux Axes Settings

Number of aux axes: 0