EtherCAT®

SCIPC可编程控制器

编程手册

软控(深圳)电气有限公司

网址:www.scatech.com.cn

地址：深圳市光明区凤凰街道尚智科园1B栋2205号

前 言

感谢您使用软控 SCIPC 可编程控制器产品。SCIPC 可编程控制器采用书本型金属机身，提供丰富的通讯总线接口来满足各种项目的扩展需求，具备强大的运动控制性能。搭载 Intel 处理器，内置 DDR4 内存，最大可扩展至 32GB，内置可扩展 SSD 固态硬盘，非常适合高负荷运算应用。本手册面向 SCIPC 可编程逻辑控制器,主要介绍 PLC 编程软件 Codesys 对控制器编程时所需的网络配置、编程环境、编程语言、程序诊断等相关知识。本公司保留对产品不断改进的权利，资料版本请以公司网站(www.scatech.com.cn)最新公布为准，恕不另行通知。

修改记录

由于产品版本升级或其他原因，本文档会不定期更新，恕不另行通知。

版本信息	
日期	版本号
2025-03-16	V1.0

感谢您购买使用软控（深圳）电气有限公司自主研发、生产的可编程控制器！本手册主要描述控制器模块的规格、特性及使用方法等，使用前敬请详细阅读，以便更清楚、安全地使用本产品

目　录

1警告提示系统

1.1 安全声明

1.在安装、操作、维护产品时，请先阅读并遵守以下安全注意事项。

2.为保障人身和设备安全，在安装、操作和维护产品时，请遵循产品上标识及手册中说明的所有安全注意事项。

3.手册中的“注意”、“警告”和“危险”事项，并不代表所应遵守的所有安全事项，只作为所有安全注意事项的补充。

4.本产品应在符合设计规格要求的环境下使用，否则可能造成故障，因未遵守相关规定引发的功能异常或部件损坏等不在产品质量保证范围之内。

5.因违规操作产品引发的人身安全事故、财产损失等，我司将不承担任何法律责任。

1.2 安全等级定义

如果不按规定操作，则可能导致死亡或严重身体伤害。

如果不按规定操作，则可能导致轻微身体伤害或设备损坏。

请妥善保管本指南以备需要时阅读，并请务必将本手册交给最终用户。

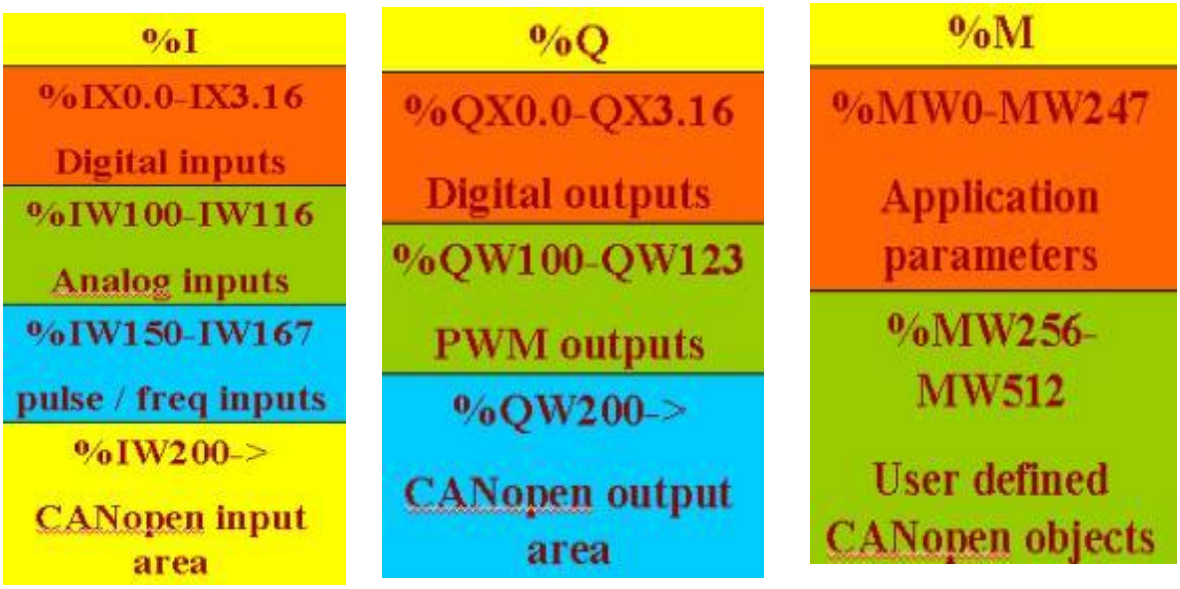
1.3 控制系统设计时安全注意事项

“警告”
● 请务必设计安全电路，保证当外部电源掉电或可编程控制器故障时，控制系统依然能安全工作； ● 超过额定负载电流或者负载短路等导致长时间过电流时，模块可能冒烟或着火，应在外部设置保险丝或断路器等安全装置。

“注意”
● 务必在可编程控制器的外部电路中设置紧急制动电路、保护电路、正反转操作的互锁电路和防止机器损坏的位置上限、下限互锁开关； ● 为使设备安全运行，对于重大事故相关的输出信号，请设计外部保护电路和安全机构； ● 可编程控制器 CPU 检测到本身系统异常后可能会关闭所有输出；当控制器部分电路故障时，可能导致其输出不受控制，为保证正常运转，需设计合适的外部控制电路； ● 可编程控制器的继电器、晶体管等输出单元损坏时，会使其输出无法控制为 ON 或 OFF 状态； ● 可编程控制器设计应用于室内、过电压等级 II 级的电气环境，其电源系统级应有防雷保护装置，确保雷击过电压不施加于可编程控制器的电源输入端或信号输入端、控制输出端等端口，避免损坏设备。

2 编程须知

2.1 控制器寄存器地址



地址类型：

Input： %I Output:： %Q Marker： %M

大小：

X： bit W： word B： byte D： dword

%IX0.0...IX0.15=%IW0，

%QX0.0...QX0.15=%QW0

， %IW0=%IB1+%IB2，

%QW0=%QB1+%QB2，

%MW0-%MW247 为248 个应用参数。

例： %IX0.0-IX3.16 为开关量输入， %IW100-IW116 为模拟量输入。

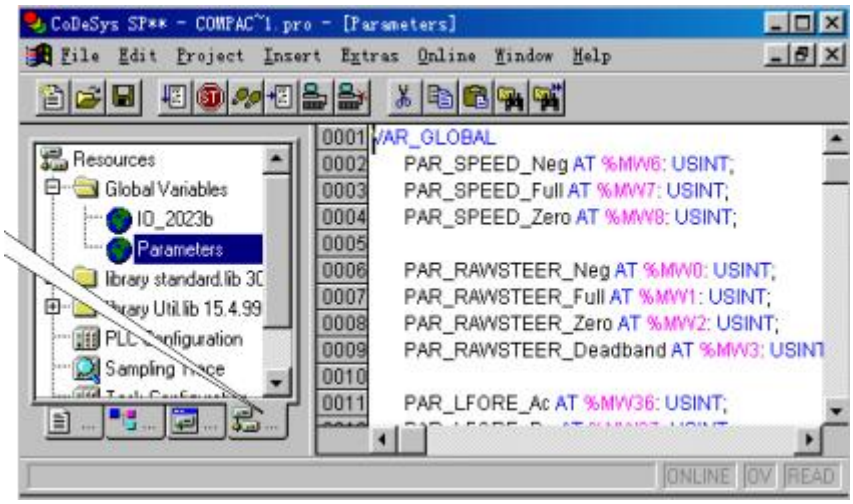
2.2 程序结构

一个工程文件包含 PLC 程序里的所有对象：POUS(program organization units)、数据类型、资源。POUs 包括主程序（PRG）、子程序（PRG）、功能块（FB）、函数（FUN）及语句。

- 主程序必须命名为 PLC_PRG。

在工程文件中，按适用范围有两种类型的变量，全局变量（Global）、局部变量(local)。全局 变量存在于程序的任何模区域，而局部变量只存在于子程序，函数和功能块中。全局变量的说明 在“resource”的 “global veriable”里：

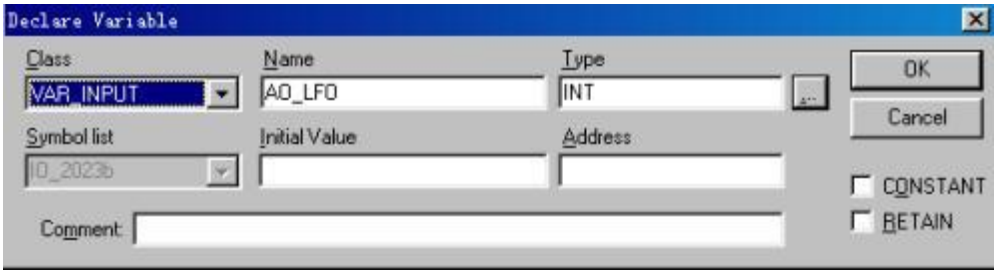
resource



局部变量的说明在“程序体”上部的局部变量说明区。

变量说明有两种方式： 一种是在变量区进行说明；另一种是自动说明。自动说明是在主菜单 里选择“Project”，“Option”，“Edit”，出现以下对话框：

选中“Autodeclaration”。这样，当编写程序，写到新的变量时，自动弹出对话框：



输入要定义的变量类型、地址、初始值。局部变量不用指定地址。

3 编程语言及数据类型

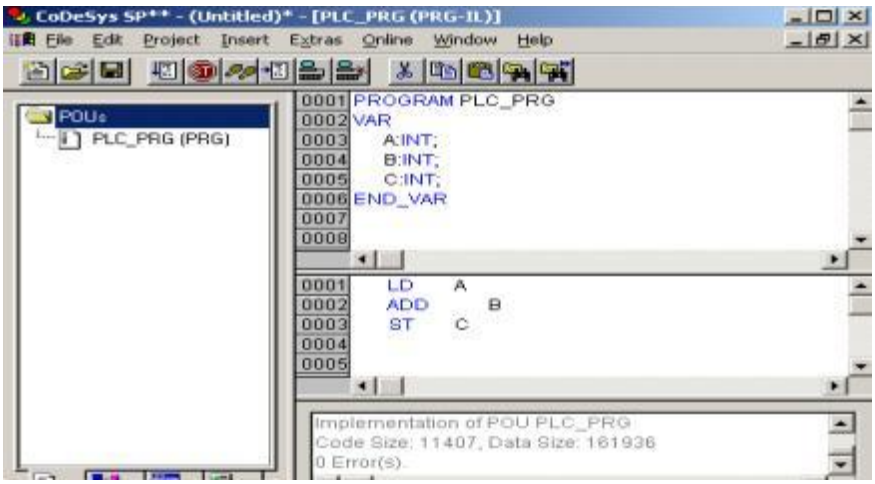
CoDeSys 是一种功能强大的 PLC 软件编程工具，它支持 IEC1131-3 标准 IL、ST、FBD、LD、 SFC、CFC 六种PLC 编程语言，用户可在同一项目中根据需要选择不同的语言编写子程序、功能 模块等。

3.1 指令表 IL

IL 程序设计语言是用布尔助记符来描述程序的一种程序设计语言。

指令	描述
LD	读取操作数的值；
ST	把当前值存入操作数；
S	把布尔量操作数置 TRUE；
R	把布尔量操作数置 FALSE ；
AND	逻辑与；
OR	逻辑或；
NOT	逻辑非；
XOR	逻辑异或；
ADD	加；
SUB	减；
MUL	乘；
DIV	除；
MOD	求余数；
GT	>；
GE	>=；
EQ	=；
NE	<>；
LE	<=；
LT	<；
JMP	跳转至标志行；
CAL	调用子程序；
RET	返回主程序；

例：

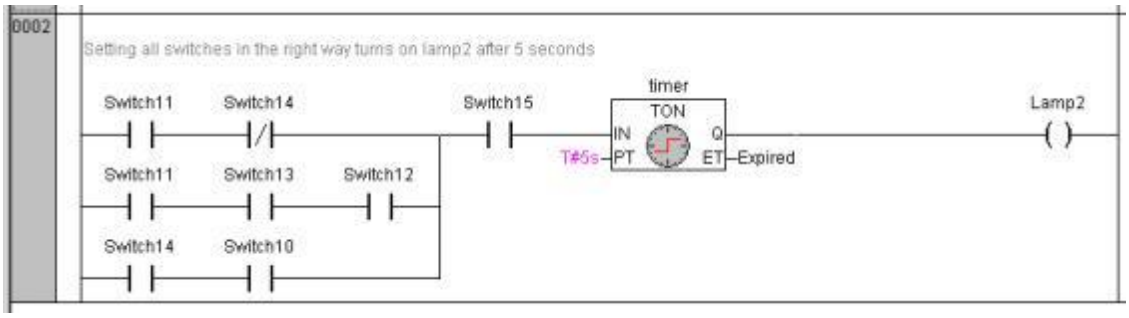


其余指令参见 CoDeSys 手册,例：

```
LD      TRUE        (*读取  TRUE*)
ANDN    BOOL1       (*与变量  BOOL1 的反值进行 AND  运算*)
JMPC    mark        (*当结果为  TRUE 时,跳转至  "mark"行*)
LDN     BOOL2       (*变量 BOOL2 值取反*)
ST      ERG         (*把结果存至 ERG*)
JMP     end         (*跳转至  "end"行*)
mark:               (*"mark"标志行*)
LD      BOOL2       (*读取 BOOL2 的值   *)
ST      ERG         (*把 BOOL2 存至 ERG*)
End:                (*"end"标志行*)
```

3.2 梯形图

梯形图程序设计语言是用梯形图的图形符号来描述程序的一种程序设计语言。这种程序设计 语言采用因果关系来描述事件发生的条件和结果。每个梯级是一个因果关系。在梯级中，描述事 件发生的条件表示在左面，事件发生的结果表示在后面。它来源于继电器逻辑控制系统的描述。 指令参见 CoDeSys 手册，例：



梯形图程序设计语言的特点是：

与电气操作原理图相对应，具有直观性和对应性；主要应用于开关量逻辑控制目的的程 序。

与原有的继电器逻辑控制技术的不同点是，梯形图中的能流（Power FLow）不是实际意 义的电流，内部的继电器也不是实际存在的继电器，因此，应用时，需与原有继电器逻辑控 制技术的有关概念区别对待。

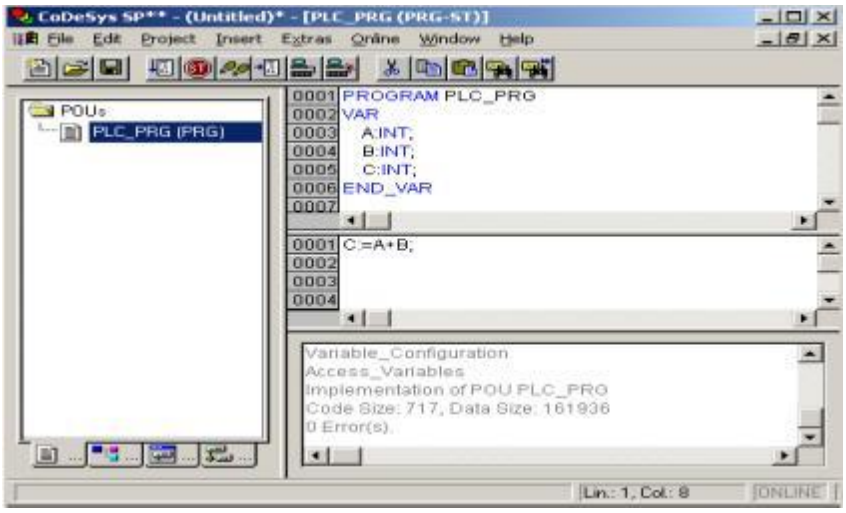
与指令表程序设计语言有一一对应关系，便于相互的转换和程序的检查。

3.3 结构化文本 ST

ST 程序设计语言是用结构化的描述语句来描述程序的一种程序设计语言。它是一种类似于高 级语言的程序设计语言。在大中型的可编程序控制器系统中，尤其是大量的模拟量运算和处理， 常采用 ST 语言来描述控制系统中各个变量之间较复杂的控制运算关系，完成所需的功能或操作。

ST 语言与BASIC语言或C语言等高级语言相类似， 但为了应用方便，在语句的表达方法 及语句的种类等方面都进行了简化。

指令	描述
AND	逻辑与；
OR	逻辑或；
NOT	逻辑非；
XOR	逻辑异或；
+	加；
-	减；
*	乘；
/	除；
MOD	求余数；



ST 语言除有一般算术运算、逻辑运算、表达式、调用子程序等指令外，还有一些结构化的语句模块。

条件语句：

```
IF        <逻辑表达式 1> THEN          (*表达式为TRUE 时，执行功能块一*)
...                                    (*功能块一*)
ELSIF <逻辑表达式 2> THEN              (*表达式为TRUE 时，执行功能块二*)
...                                    (*功能块二*)
ELSIF <逻辑表达式 n> THEN              (*表达式为TRUE 时，执行功能块n*)
...                                    (*功能块 n*)
ELSE
...                                    (*功能块 n+1*)
END_IF                                 (*  条件结束*)
```

其余指令用法参见 CoDeSys 手册。

3.4 功能模块图 FBD

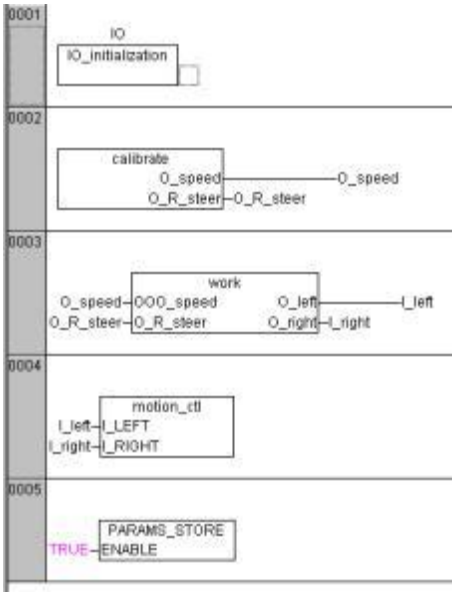
功能模块图程序设计语言是采用功能模块来表示模块所具有的功能，不同的功能模块有不同,它有若干个输入端和输出端，通过软连接的方式，分别连接到所需的其它端子，完成所 需的控制运算或控制功能。功能模块可以分为不同的类型，在同一种类型中，也可能因功能参数 的不同而使功能或应用范围有所差别，例如，输入端的数量、输入信号的类型等的不同使它的使 用范围不同。由于采用软连接的方式进行功能模块之间及功能模块与外部端子的连接，因此控制 方案的更改、信号连接的替换等操作可很方便实现。功能模块图程序设计语言的特点是：

以功能模块为单位，从控制功能入手，使控制方案的分析和理解变得容易；

功能模块是用图形化的方法描述功能，它的直观性大大方便了设计人员的编程和组态，有较 好的易操作性；

对控制规模较大、控制关系较复录的系统，由于控制功能的关系可以较清楚地表达出来，因 此，编程和组态时间可以缩短，调试时间也能减少；

用户自己可以用不同的语言编写特定的功能模块，也可在用其他语言编程时插入功能模块。 例：



上例中，共有五个模块，分别实现 PLC 初始化、输入模拟量标定、工作控制、PWM 输出、 参数 FLASH 等五种功能。

3.5 顺序流程图 SFC

SFC 语言是用顺序流程图来描述程序的一种程序设计语言。采用顺序流程图的描述，控制系 统被分为若干个子系统，从功能入手，使系统的操作具有明确的含义，便于设计人员和操作人员 设计思想的沟通，便于程序的分工设计和检查调试。

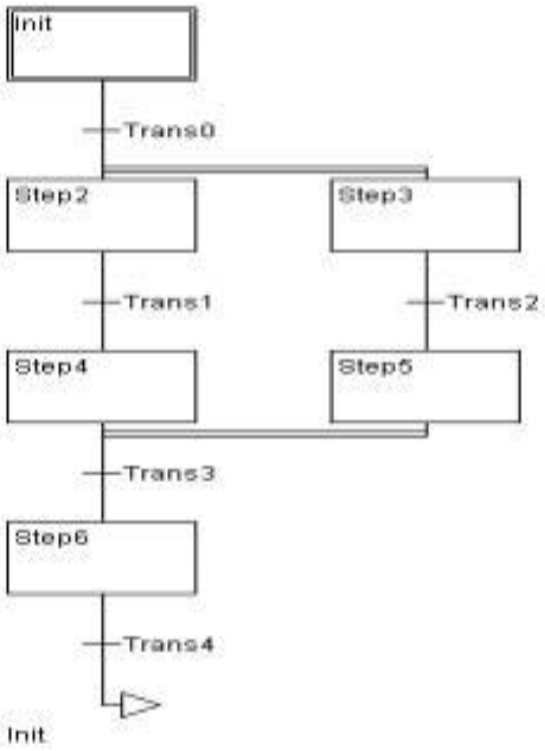
SFC 语言的特点是：

以功能为主线，条理清楚，便于对程序操作的理解和沟通。

对大型的程序，可分工设计，采用较为灵活的程序结构，可节省程序设计时间和调试时间。

能较简单和清楚地描述并发系统和复杂系统的所有现象，并能对系统中存有的死锁、不安全 等反常现象进行分析和建模，在模型的基础上能直接编程 。

例：



当且仅当顺序条件为真时，程序才能往下执行。每一步功能块可用不同语言编写，同等条件的功能块，可指至时间执行顺序

3.6 数据类型

在 CoDeSys 环境中，有以下标准数据类型：

- BOOL（布尔量）
- SINT（短整型） 、INT（整型数） 、DINT（双整型数）
- USINT（无符号短整型） 、UINT（无符号整型数） 、UDINT（无符号双整型数）
- BYTE（位）、WORD（字） 、DWORD（双字）
- STRING（字符量）
- REAL（实型数） 、LREAL（长实型数）
- TIME（时间量） 数值取值范围：

Type	下限	上限	存储空间
BYTE	0	255	8 Bit
WORD	0	65535	16 Bit
DWORD	0	4294967295	32 Bit
SINT	-128	127	8 Bit
USINT	0	255	8 Bit
INT	-32768	32767	16 Bit
UINT	0	65535	16 Bit
DINT	-2147483648	2147483647	32 Bit
UDINT	0	4294967295	32 Bit

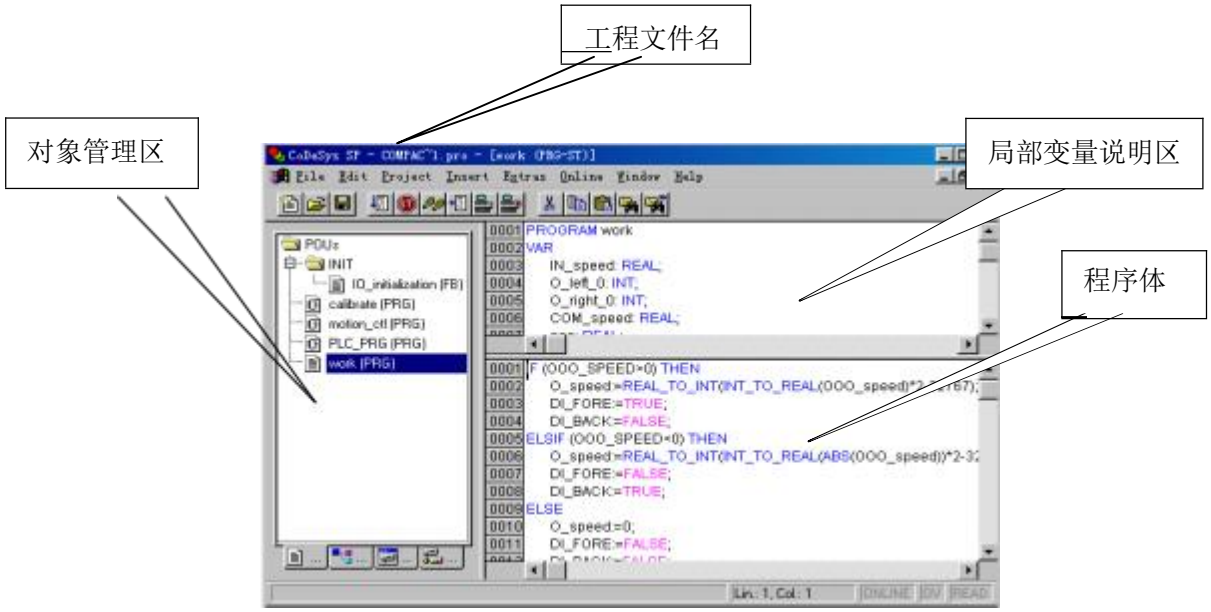
2、自定义数据类型：

自定义数据类型有数组、指针、结构等，具体操作参见 CoDeSys 手册。

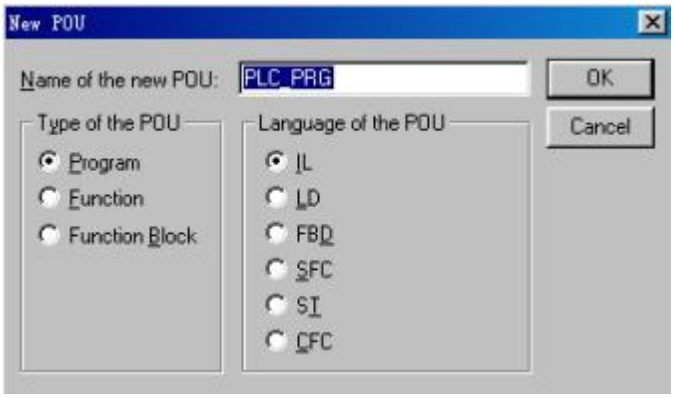
4 程序的创建

4.1 程序结构

一个 POU 由两部分组成：变量说明部分和程序体。例：



创建一个工程文件时，打开 CoDeSys 后选择\主菜单\File\New，这时出现一个对话框：



这是此工程文件的第一个 POU，已经命名为一个默认的名字 PLC_PRG，类型被定义为程序。 不要改动名字及类型，选择一种要编程的语言，创建第一个 POU。因为在任何一个工程文件中，必须要有这一文件，一般地，把它作为主程序。

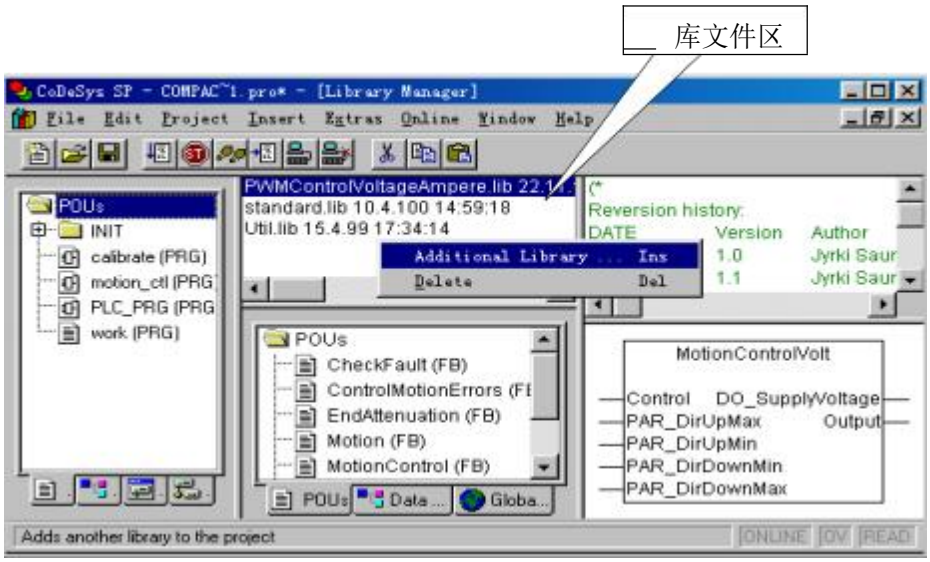
要创建其他的 POU 时，把光标移到对象管理区，按鼠标右键，选择“Add Object”,或选择主菜单 “Project” “Object” “Add”，出现上述菜单，输入文件名、选择 POU 类型及编程语言，按“OK” 结束。

当你存盘时，系统会提示你输入工程文件名，文件名的命名原则跟 Windows 文件名要求一致。

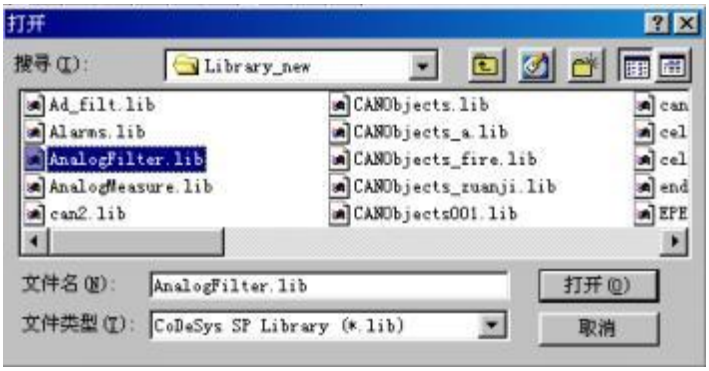
4.2 调用库文件

CoDeSys 有大量的库文件（*.lib）供用户编程时调用。当你要调用某一操作指令时，把包含 该指令的库文件调入当前工程文件的库中。操作如下：

点击主菜单“Window”“Library Manager”,弹出对话框：



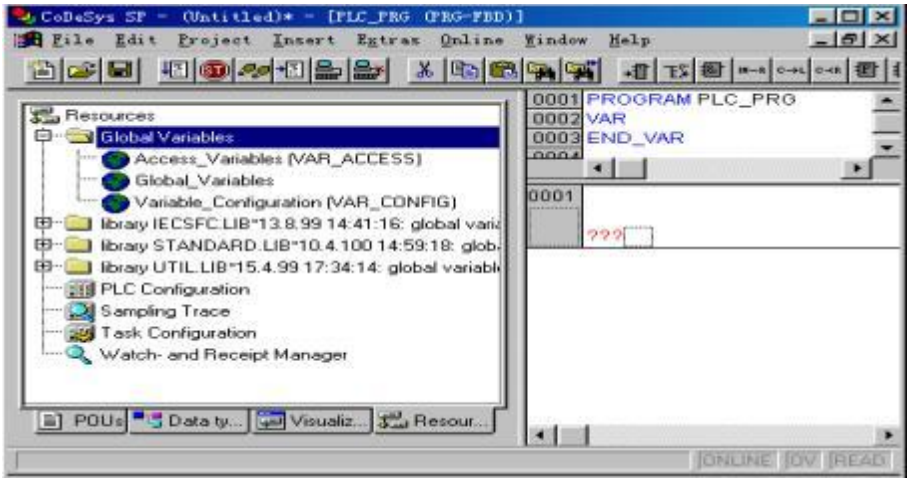
将光标移到库文件区，按右键，选择“Additional Library …Ins”,弹出对话框：



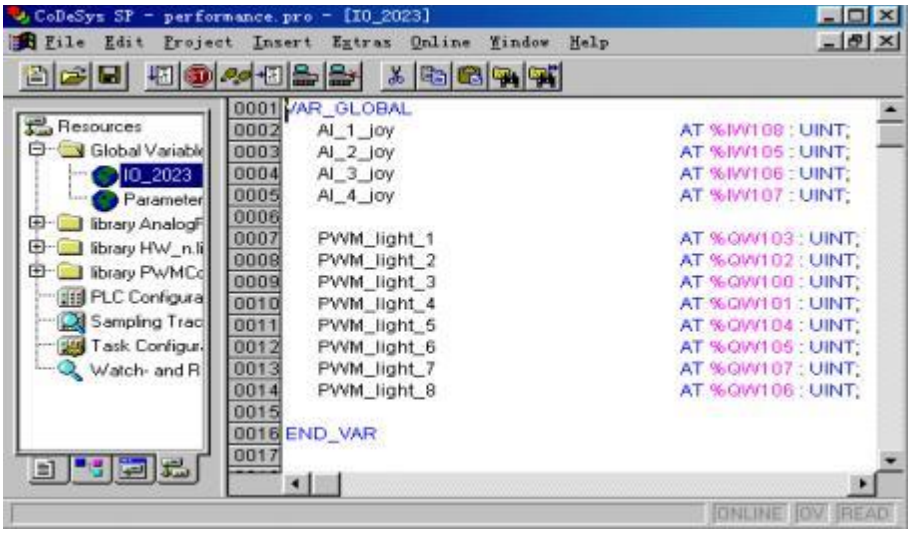
选中要用的文件即可。关于一些库文件的详细说明，请参看下一章。

4.3 程序实例

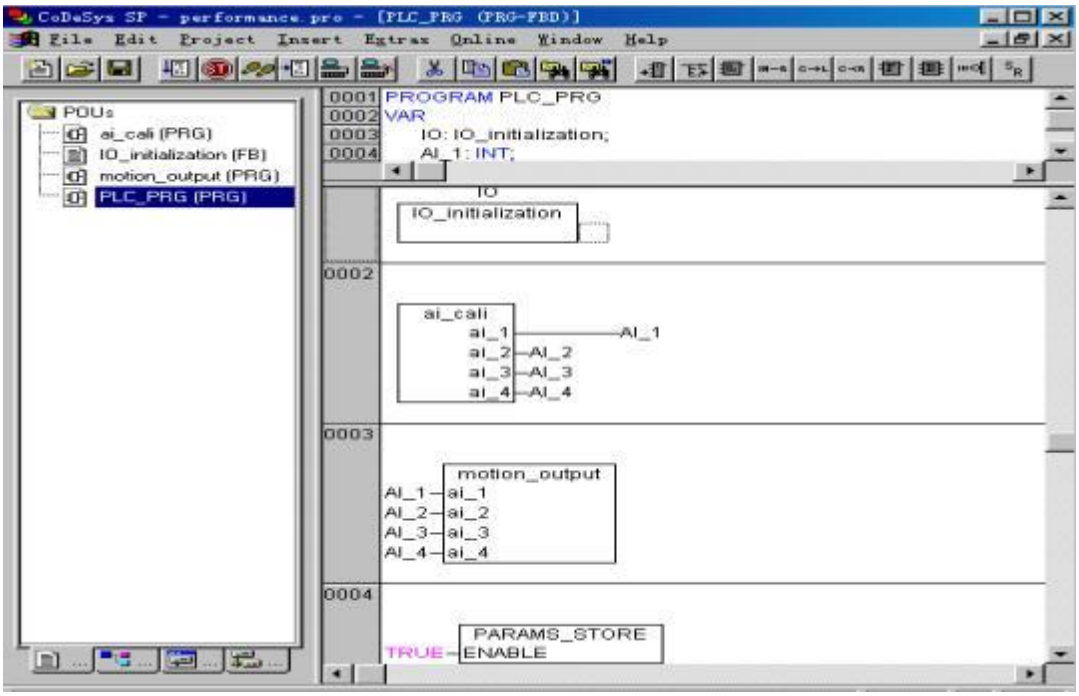
这里，以演示台的例子来说明程序建立的过程。该演示台用两个万向手柄控制八个灯泡，每 个灯泡对应一个手柄某一方向，手柄位移的大小控制灯泡的亮度。本例采用 2023 控制器。首先， 编写项目的控制方案和程序流程，对控制器的 I/O 口进行分布。按前面的方法建立工程文件取名 Performance，并以 PLC_PRG 为主程序，采用 FBD 语言。接下来设置I/O 口，点击对象管理取的 “Resources”按扭，弹出对话框：



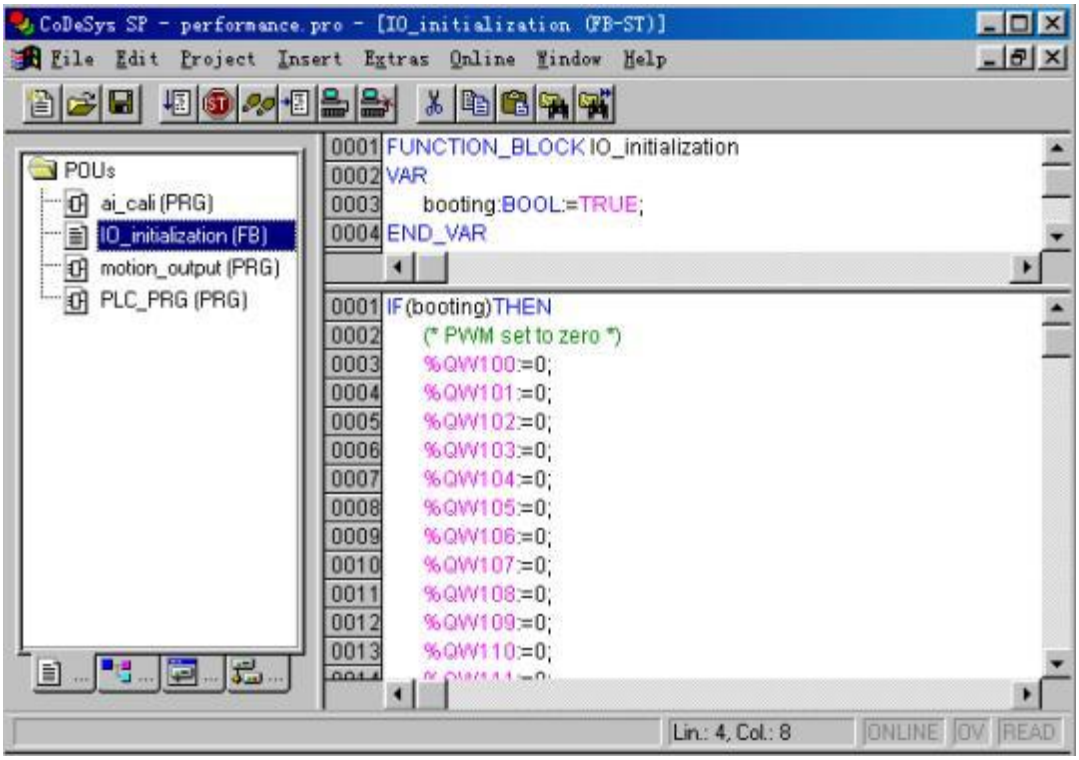
把 “Global Varibels”下面的“Access_Varibles”更名为“IO_2023”， “Global_Varibles”更名为 “Parameter”。双击 “IO_2023”,设置 I/O 口，如下图：



在主程序中，先要对 I/O 进行初始化，编写 I/O 初始化模块，并在主程序的第一模块调用该模 块，然后编写其他程序模块。主程序模块如下：



I/O 初始化程序如下：



在此程 序中，“ ai_cali” 模块为手柄输入标定， “ motion_output” 为 PWM 信号输出，“PARAMS_S～RE”为参数固化模块。

5 指令操作

5.1 指令表

指令表中包含一系列的指令，依赖于操作的类型,每一条指令在一个新行开始并且包含运算符号和一 个或多个用逗号隔开的操作数。

在一个指令前面，还可以有一个标号，后缀一个冒号。

注释部分在一行的最后，指令与指令之间可以插入空行。 例如：

LD 17

ST lint (* Kommentar *)

GE 5

JMPC next LD idword

EQ istruct.sdword

STN test next:

请参考：

指令表中的修饰符和操作符

指令表中的修饰符和操作符

在指令列表中将用到下面的操作符和修饰符： 修饰符：

• C	与操作符 JMP, CAL, RET 连用:	当前面的表达式处理的结果为 TRUE 时，才执 行此指令。
• N	与 操 作 符 JMPC, CALC,RETC 连用:	当前面的表达式处理的结果为 FALSE 时，才 执行此指令。

•N	用于其它情况:	取操作数的反 (不包括累加器)。

下面是操作符和它们可能的修饰符以及相关的意思：

操作符	修饰符	含意
LD	N	使当前的值等于操作数
ST	N	在操作数的位置保存当前值
S		当当前的值为 TRUE 时，把布尔型操作数置为 TRUE
R		当当前的值为 TRUE 时，把布尔型操作数置为 FALSE
AND	N,(	位逻辑运算符号“与”
OR	N,(	位逻辑运算符号“或”
XOR	N,(	位逻辑运算符号“异或”
ADD	(	加法
SUB	(	减法
MUL	(	乘法
DIV	(	除法
GT	(	>
GE	(	>=
EQ	(	=
NE	(	<>
LE	(	<=
LT	(	<
JMP	CN	跳转到标号
CAL	CN	调用程序功能块
RET	CN	离开 POU 并返回到调用的地方
)		执行延时操作

单击这里可以得到所有 IEC操作符的列表。 使用修饰符编写的程序的例子：

LD	TRUE	(*把 TRUE 加载到累加器中*)
AND N	BOOL 1	(*执行 AND 和 BOOL1 变量的反之“与”*)
JMPC	mark	(*当上面的结果为 TRUE 时，跳转到标号“mark”处*)
LDN	BOOL 2	(*保存 BOOL2 的反 *)
ST	ERG	(*把 BOOL2 保存在 ERG*)
标号:		
LD	BOOL 2	(*保存 BOOL2 的值 *)
ST	ERG	(*把 BOOL2 保存在 ERG*)

在 IL 中也可以在操作之后放一个圆括号。圆括号内的值被认为是一个操作数。 例如：

```
LD  2 MUL 2 ADD 3 Erg
```

这里 Erg 的值为 7，但是如果加一个圆括号：

```
LD 2
MUL (2 ADD 3  )
ST Erg
```

Erg 的结果是 10，当到达")"时操作 MUL 才开始计算；此时对操作数计算 MUL 5。

5.2 结构化文本

结构化文本中包含一系列的指令 ， 这些用高级语言编写的指令能够被执行 （例如 IF……THEN……ELSE）或者在循环（WHILE..DO）。

例如：

```
IF value < 7 THEN  WHILE value < 8 DO value:=value+1;
END_WHILE;
END_IF; 参照：
```

表达式

表达式的计算 对操作数赋值

在ST中调用功能块 RETURN 指令

IF 指令

CASE 指令

FOR 循环

WHILE 循环 REPEAT 循环 EXIT 指令

表达式

表达式是一个在运算后返回一个值的结构。

表达式由运算符和操作数组成，操作数可以是常量、变量、功能调用或其它表达式。

表达式的计算

依照一定的规则来处理运算符号可以计算出表达式的值，约束力最高的运算符首先参加运算，然后 是约束力稍高的运算符，直到所有的运算符都被处理为止。

运算符号“=”号的处理是从左到右的顺序。

下面是结构文本中运算符号约束力的级别排列

操作	符合	约束力
放入圆括号	(表达式)	最强的约束力
功能调用	Function name (parameter list)	
求幂	EXPT	
取反	NOT	

乘法 除法 取模	* /	

	MOD	
加法 减法	+ -	
比较	<,>,<=,>=	
等于 不等于	= <>	
布尔运算与	AND	
布尔运算异或	XOR	
布尔运算或	OR	最弱的约束力

下面这些是结构化文本中的其它指令，和例子一起安排在一个表中。

指令类型	例子
赋值	A:=B; CV := CV + 1; C:=SIN(X);
调用一个功能块并使 用功能块输出	CMD_TMR(IN := %IX5, PT := 300); A:=CMD_TMR.Q
RETURN	RETURN;
IF	D:=B*B; IF D<0.0 THEN C:=A; ELSIF D=0.0 THEN C:=B; ELSE C:=D; END_IF;
CASE	CASE INT1 OF 1: BOOL1 := TRUE; 2: BOOL2 := TRUE; ELSE BOOL1 := FALSE;

	BOOL2 := FALSE; END_CASE;
FOR	J:=101; FOR I:=1 TO 100 BY 2 DO IF ARR[I] = 70 THEN J:=I; EXIT; END_IF; END_FOR;
WHILE	J:=1; WHILE J<= 100 AND ARR[J] <> 70 DO J:=J+2; END_WHILE;

REPEAT	J:=-1; REPEAT J:=J+2; UNTIL J= 101 OR ARR[J] = 70 END_REPEAT;
EXIT	EXIT;
空指令	;

对操作数赋值

“=”号左边是一个操作数（变量，地址），它的右边是赋予它的表达式的值 例如：

Var1 ：=Var2*10

在运算结束后，变量 Var1 就得到了 Var2 的 10 倍值。

在结构化文本中调用功能块

通过写 功能块的实例名和随后在括号中给参数分配值来调用一个 功能块 ，在下面的例子中，通过 给两个参数IN和PT赋值来调用一个定时器，然后结果变量Q的值赋予变量A

结果变量，就象在指令表中，被表示为功能块名称后跟一个小点和变量的名字。 CMD_TMR(IN := %IX5, PT := 300);

A:=CMD_TMR.Q

RETURN 指令

返回指令可以用来按照条件离开一个 POU（程序组织单元）。

IF 指令

IF 指令可以检验一个条件，根据这个条件，执行指令。 语法：

IF <Boolean_expression1> THEN <IF_instructions>

{ELSIF <Boolean_expression2> THEN <ELSIF_instructions1>

ELSIF <Boolean_expression n> THEN <ELSIF_instructions n-1>

ELSE

<ELSE_instructions>} END_IF;

在{}中的部分是可选的。

如果布尔运算表达式<Boolean expression>返回 TRUE，只有 if 指令部分执行，其它部分不执行。

否则，布尔运算表达式从<Boolean expression 2>开始，一个接一个的计算，直到某个布尔表达式 返回为 TRUE，然后，在这个布尔运算表达式 2 之后，ELSE 或 ELSE IF 之前的部分被计算。

如果没有任何一个布尔运算表达式返回 TRUE，那么只计算 ELSE 下的指令 例如：

IF temp<17

THEN heating_on := TRUE; ELSE heating_on := FALSE; END_IF;

这里当温度降到 17 度以下时加热开始，否则保持关闭状态。

CASE 指令

18

使用 CASE 指令，可以在一个结构中，用同一个条件变量组合多个有条件的指令。

句式：

CASE <Var1> OF

<Value1>: <Instruction 1> <Value2>: <Instruction 2>

<Value3, Value4, Value5>: <Instruction 3> <Value6 .. Value10>: <Instruction 4>

...

<Value n>: <Instruction n> ELSE <ELSE instruction>

END_CASE;

•CASE 指令根据下面的模式来处理

•如果变量 Var1 有值 Value1，那么执行指令 Instruction1。

•如果变量 Var1 不是所指明的值，那么执行 ELSE Instruction。

•如果有多个变量值要执行同一个指令，那么这些条件执行一个公共指令

•如果对于一个变量在一个值的范围内执行同一个指令，那么在初始值和最后值之间用两个句点隔 开，所以你可以规定公共条件。

例如：

```
CASE INT1 OF
1, 5: BOOL1 := TRUE;  BOOL3 := FALSE;
2: BOOL2 := FALSE;  BOOL3 := TRUE;
10..20: BOOL1 := TRUE;  BOOL3:= TRUE;
ELSE
BOOL1 := NOT BOOL1;
BOOL2 := BOOL1 OR BOOL2;  END_CASE;
```

FOR 循环

通过 FOR 循环程序可以编写重复执行的过程。 句式：

```
INT_Var :INT;
FOR <INT_Var> := <INIT_VALUE> TO <END_VALUE> {BY <Step size>} DO
<Instructions> END_FOR;
```

{}内的部分是可选的。

只要计数器 INT_Var 不大于 END_VALUE，指令 Instructions 就一直执行，在执行 Instructions 之 前首先检查计数器的值，如果 INIT_VALUE 比 END_VALUE 大的话 Instructions 将不在执行。

当 Instructions 执行后，INT_Var 通常要增加一个 Step size，Step size 可以是任何整型值，如 果没有 Step size，它将设置为 1，当 INT_Var 大到一定值时，循环结束。

例如：

```
FOR Counter:=1 TO 5 BY 1 DO Var1:=Var1*2;
END_FOR;
Erg:=Var1;
```

- -

我们假设 Var1 的默认值是 1，那么在循环结束后它将得到值 32

注意：END_VALUE 一定不要大于等于与计数器 INT_VAR 的极限值，例如：如果变量计数器是一个 SINT 类型并且 END_VALUE 为 127，那么这将是一个死循环。

WHILE 循环

WHILE 循环可以象 FOR 循环那样使用，不同之处在与 WHILE 循环的退出条件可以是任何布尔型表达 式，当条件满足时，就会执行循环。

句式：

```
WHILE <Boolean expression>
<Instructions> END_WHILE;
```

只要 Boolean_expression 返回 TRUE，那么就重复执行 Instructions 如果 Boolean_expression 在 首次计算出 FALSE，那么指令将不再执行，如果 Boolean_expression 从不出现 FALSE，Instructions 将 没完没了的重复执行。

注意：程序员必须保证不出现死循环，这可以通过改变循环中指令部分的条件来实现，例如：可以通过计数器增加 或减少。

例如：

```
WHILE counter<>0 DO Var1 := Var1*2;
Counter := Counter-1;  END_WHILE
```

对于 WHILE 和 REPEAT 个循环在循环之前不必知道循环的次数，从这个意义上来说，这两种循环要比 FOR 要强大一些。因此在这种情况下，可以用这两种循环。如果循环数比较明确，那么 FOR 循环因为没 有死循环而更好一点。

REPEAT 循环

REPEAT 循环和 WHILE 循环的不同之处在于它的中断条件是在循环执行之后才被检查，这就是说，循 环至少要执行一次，不管中断是什么条件

句式： REPEAT

<Instructions>

UNTIL <Boolean expression> END_REPEAT;

Instructions 一直执行到 Boolean expression 返回 TRUE

如果 Boolean expression 第一次就赋予真值，Instructions 只执行一次，否则 Instructions 将重 复执行将会导致时间延迟。

注意：程序员可以通过改变循环中指令部分的条件来保证没有死循环出现，例如：可以通过计数器增加或减少。 例如：

```
REPEAT
Var1 := Var1*2
Counter := Counter-1;
UNTIL
Counter=0
END_REPEAT;
```

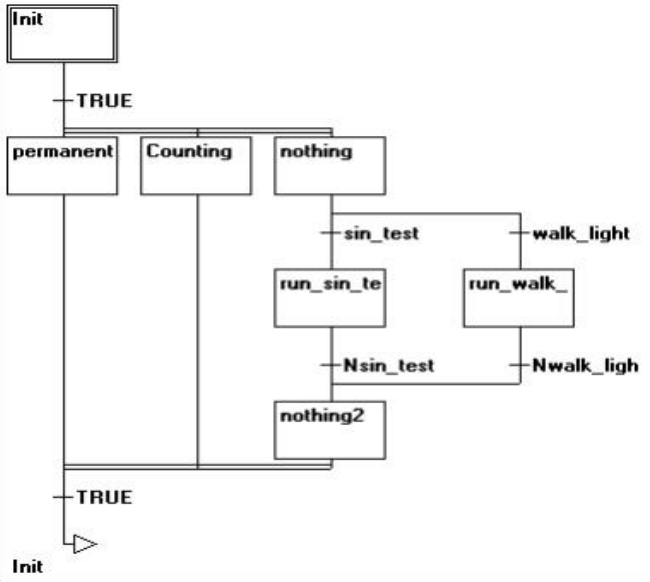
EXIT 指令

如果在 FOR WHILE 或 REPEAT 循环中有 EXIT 指令，那么内循环就结束，不管中断是什么条件。

5.3 顺序功能图（SFC）

顺序功能图是基于图形化的语言，用它可以描述一个程序中不同动作的先后顺序。因为这些动作分 配给单步元素，通过变迁元素来控制处理的顺序。

下面是一个顺序功能图的例子：



关于顺序功能图编辑器的更多知识请参照 5.4.4 章节 参照：

- 步
- 动作
- 进入和退出动作 转换/转换条件
- 激活步 IEC 步 限定词
- 顺序功能图种的隐含变量
- SFC 标志符 可选分支
- 平行分支 跳转

在联机模式下可参考编辑和行为信息：

- 顺序功能图编辑器
- 顺序功能图联机模式
- 步

用顺序功能图编写的程序组织单元包含了一系列的步，这些步之间是通过定向连接（转换条件实现的。

有两种类型的步：

- 简单类型：每步包括一个 动作 和一个标记，这个标记用来表示此步是否激活。如果单步动作正在 执行，那么在步的右上角方向会出现一个小三角形。
- IEC 类型：每步包含一个标记和一个或多个赋值的动作或布尔变量。相关的动作出现在步的右边。

动作

一个动作可以包含一系列的指令表或结构化文本指令，功能模块图或 梯形图许多的网络，或者又包 含另外顺序功能图。

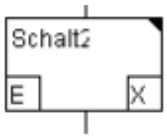
在 简单步 中，动作经常是和步连接在一起的，为了能编辑一个动作，在步上双击鼠标或选择此步 再选择菜单命令 ’扩展’ ’快速动作/转换’。另外，每一个步中允许一个输入或输出动作。

IEC步的动作是附加在顺序功能图-程序组织单元内的对象管理器中，通过双击或者在它的编辑器中 按Enter键可以加载它。也可以通过“’工程’ ’添加动作’来创建一个新的动作。你可以为一个IEC步分配 最多九个动作。

进入和退出动作

可以额外的为一个步添加一个进入和退出的动作，在一个步激活后，一个进入动作只能执行一次。 退出动作只在步失效之前执行一次。

进入动作用左下角一个“E”来表示，退出动作用右下角的“X”表示。 下面是一个带有进入和退出动作的步的例子：



转换/转换条件

在步和步之间有所谓的转换。

转换条件的值必须是TRUE或FALSE，因而它可以是一个布尔变量、布尔地址或布尔常量。在结构化文 本句式（例如（I<=100）AND b）或者在任何一种期望的语言（参照 ’附加’ ’快速动作/转换’）中，它 也能包括一系列有布尔结果的指令。转换中不能包括程序、功能块或赋值。

注意：除了转换外，也能用渐进模式跳到下一步，查看 SFCtip 和 SFCtipmode

激活步

在调用顺序功能图的POU后，初始化步的 动作（被一个双边线包围）将首先执行。动作正在执行的 步，状态是激活的，在联机模式下，激活的步显示为蓝色。

在一个控制循环中激活步的所有动作都将执行。所以，当激活步之后的 转换条件是TRUE时，它之后 的步被激活。当前激活的步将在下个循环中再执行。

注意：如果激活的步包含一个输出动作，譬如它下面转换条件是 TRUE，那么它只能在下个循环过程中执行。

IEC 步

在顺序功能图中可以使用标准的 IEC 步。

为了能使用 IEC 步，你必须在你的工程文件中联接 Iesfc.lib 库文件。

一个IEC步中不能分配超过九个 动作 ，IEC的动作不象简单步那样固定地作为输入或输出到某个步 的动作，而是和步分开存储并且能够在一个程序组织单元中重复使用多次。因此，它们必须用命令‘扩 展连接动作’和单个步联系在一起。

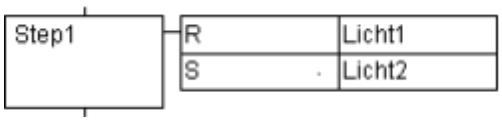
除了动作，布尔变量也能分配给步。

能够使用所谓的 限定词来控制激活和未激活的动作和布尔变量。时间延迟是可能的， 如果一个动作 依然激活这，而下一个步已经开始处理了。通过限定词S（设置），可以取得并发的过程。

随着每一个顺序功能模块的调用，相关联的布尔变量被设置或复位，也就是说，随着每一次调用， 这个值将在 TRUE 到 FALSE 之间来回变化。

IEC 步的关联动作在步右边的两长方形中表示，左边的区域包含了限定词，可能带有时间常量，右边的区域包含了动作名和各自的布尔变量名。

下面是一个带有两个动作的 IEC 步：



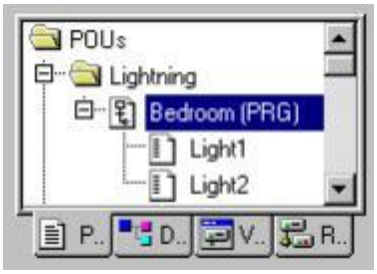
为了处理的方便，联机模式下的所有激活动作象激活步一样都显示为蓝色，在一个循环之后检查一 次哪个动作是激活的。

注意：如果一个动作已经失去激活了。它会再执行一次，这就是说，每一个动作至少被执行两次。 在首次调用一个未激活的动作时，激活的动作将按字母表的顺序执行。

一个新插入的步是不是 IEC 步，取决于命令菜单 ‘’扩展’’使用 IEC 步’是否被选中。

在对象管理器中，动作都直接存放在各自的SFC POU中，新的动作可以通过 ’工程’’添加动作’.来 创建。

要使用 IEC 步，你必须在工程文件中包含特殊的 SFC 库文件 lecsfc.lib 在对象管理器中带有动作的SFC POU



限定词

为了关联 动作和 IEC步，用到下面的限定词。

N 非存储 动作和步一起激活

R 复位 动作是未激活的

S 设置 动作被激活再复位前保持激活状态

L 时间限制 动作激活一段时间，最大和步激活时间一致

D 时间延迟 如果步仍然激活，动作在一定时间后激活，然后只要步是激活的，它就保持激活。

P 脉冲 如果步激活，动作只执行一次。

SD 存储和时间延迟 在一定时间之后动作激活并保持激活状态到下一个复位开始。

DS 延迟和保持 只要步仍然激活并且保持到下一个复位开始，那么在一定时间后动作被激活激

SL 保持和时间限制 动作激活并保持一段时间

限定词 L、 D、 SD、 DS 和 SL 需要一个 TIME 常量格式的时间值。

注意：当一个动作失去激活时，它会再执行一次。这就是说每个动作至少执行两次。

顺序功能图种的隐含变量

在 SFC 中使用一些隐含声明的变量。

每一个步都有一个标记，标记中存储着步的状态。对于 IEC步来说，步的标记（激活或未激活）被 称为<StepName>.x或者对一个简单的步来说称为<StepName>，当关联的步激活的时候这个布尔变量值为 TRUE，反之则值为FALSE。它能够用在SFC模块中的每一个IEC动作和转换中。

可以通过查询<ActionName>.x 来查询一个 IEC 步是否激活。 隐含变量<StepName>.t 能够用来查询步激活的时间。

隐含变量也能够被其它程序访问，例如，boolvar1:=sfc1.step1.x; 这里 step1.x 是隐含布尔变量， 它代表了程序组织单元 sfc1 中的 IEC 步 step1 的状态。

SFC标志符

SFC 程序组织单元标志符用来控制操作，它在工程运行期间隐含的创建，为了能读这些标志符，你 必须定义合适的全局变量或局部变量。例如，如果在一个 SFC 程序组织单元中一个步激活的时间超过了 它定义的属性，那么就会设置一个标志符，通过用一个“SFCError”变量可以访问到这个标志符（此时 SFCError 得到真值）。

可以定义下列标志符变量：

SFCEnableLimit:这个变量的类型是布尔型，当它的值为 TRUE 时，这一步的超时将会注册进 SFCError，其它的超时将被忽略。

SFCInit:当这个布尔变量值为 TRUE 时，顺序功能图复位到初始状态，其它的 SFC 标志符也会被复位。 初始步保持激活，直到变量值为 TRUE 时，才开始执行。只有当 SFCInit 被重新设置为 FALSE 时，模块才 能正常工作。

SFCReset:这个布尔变量，与 SFCInit 很相似，不同之处在于，进一步的处理发生在步的初始化之后， 因而，例如 SFCReset 标志符可以在初始化步中被复位到 FALSE。

注意:从 2.3.7.0 版编译器开始，SFCReset 可以用于复位与 IEC 步相关联的布尔型动作。

SFCQuitError:当这个布尔变量得到 TRUE 时，SFC 的执行将会停止，因此，在变量 SFCError 中一个 可能超时将复位，当这个变量呈现 FALSE 时，激活步中的所有时间都会复位，先决条件是在 SFC 中已经 定义过登记任何超时设定的标志符 SFCError。

SFCPause:当这个布尔变量值为 TRUE 时，SFC 图表的执行就会停止。

SFCError:当在 SFC 图表中有超时时这个变量得到 TRUE。如果在这个之后还有其它的超时发生，除 非是这个变量已经复位 ，否则 ，这 些状态将不会登记 。如 果你想使用其它的 时间控制标志符 (SFCErrorStep,SFCErrorPOU, SFCQuitError,SFCErrorAnalyzation)的前提条件是定义 SFCError

SFCTrans:当一个转换被驱动时，这个布尔变量得到真值。

SFCErrorStep:这个变量是一个字符变量，如果 SFCError 中登记了一个超时，这个变量将存储这个 超时步的名字。前提条件是在 SFC 中已定义了登记任何超时的标志符 SFCError。

SFCErrorPOU:这个字符变量包含了发生超时的模块名字。前提条件是在 SFC 中已定义了登记任何超 时的标志符 SFCError。

SFCCurrentStep:这个字符变量存储了那个被激活步的名字，它与的时间监控无关。在仿真的情况下， 此步存储在外部适当的分支种。如果一个超时发生其它的将不再登记，而且 SFCError 也不会复位。

SFCErrorAnalyzationTable:这是数组型变量，它提供一个转换表达式的分析结果，表达式中对转换 中的 FALSE 和上一步起时有影响的每个一元素。要把下列信息写进 ExpressionResult 结构中写进：名称， 地址，注释，当前值。

最大可以容纳 16 个元素，因此，数组的范围从（0-15）。

ExpressionResult 结构和隐含使用的分析模块都是由 AnalyzationNew.lib 库文件提供的，分析模 块也能够被其它的不用 SFC 编写的程序组织单元显式使用。

上一步的超时登记是分析一个转换表达式的先决条件，因此在这里必须有一个时间监视的应用，而 且，SFCError 必须在声明窗口被定义。

SFCTip, SFCTipMode:这个布尔变量允许 SFC 的渐进模式。当用在 SFCTipMode=TRUE 来切换它时。如 果 SFCTip 设置值为 TRUE 时它只可能跳到下一个步，只要 SFCTipMode 是设置为 FALSE 时，它可能跳过转 换。

注意:对于扫描的状况和分步运动时间 隐藏变量还是可用的。

可选分支

在SFC中可以定义两个或两个以上的可选择分支。每一个分支的开始和结束必须带有一个 转换。可 选分支可以包含平行的分支和其它的选择分支， 一个可选分支开始于一个水平线并终止于一个 水平线

（选择结束）或是一个跳跃。

如果在可选分支开始行前面的步是激活的，每一个可选分支的首次变换将从左到右被计算，最先的 转换将从左边转换条件为 TRUE 的开始，然后下面的步被激活。

平行分支

在 SFC 中可以定义两个或两个以上的分支为平行分支。每一个平行分支在开始和结束处必须有一个 步。平行分支可以包含可选择的分支或其它平行分支，一个平行分支开始于一个双划线，结束于一个双 划线或者一个跳跃，它能提供一个跳跃标识。

如果平行分支的先前步是激活的，并且这个步之后的 转换条件值是TRUE时，那么平行分支的第一步 激活。这些分支彼此平行处理。当所有平行步激活并且这些步之后转换条件为TRUE时，那么平行分之末 端线后的那一步被激活

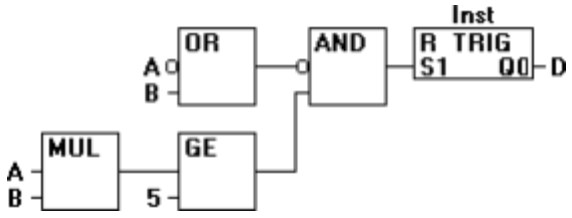
跳转

跳转是对在跳转符号下面指明的 步名的一个连接。当在不允许创建导致向上或互相交叉联系的时 候，必须使用跳转。

5.4 功能模块图

功能模块图是一种基于图形的编程语言，它用一串网络来工作，每一个网络包含一个算术或逻辑表 达式、功能块的调用、跳转或返回指令的结构。

下面是功能模块图中一个网络的例子

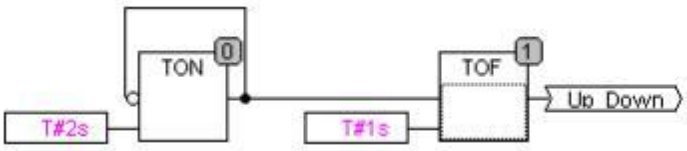


关于功能模块编辑器的更多的知识请参照 5.4.1 章节中的讲述。

5.5 连续功能图表编辑器

连续功能图表编辑器不象功能模块图表那样操作，但是可以自由放置元素，它允许使用反馈。 关于连续功能图的更多信息请参照 5.4.4 章节

连续功能图编辑器中网络的例子：



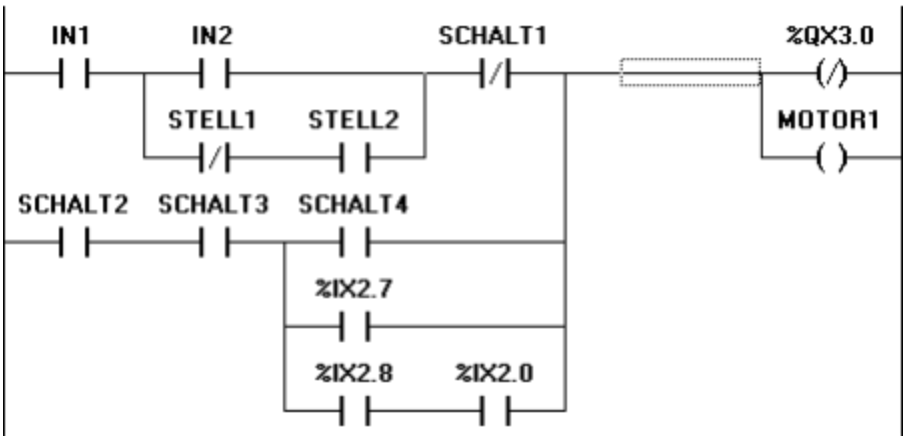
5.6 梯形图

梯形图也是一种基于图形化的编程语言，它接近于电子电路的结构，一方面，梯形图很适合构建逻 辑开关，另一方面，它也能创建象 FBD 中的网络图，所以梯形图在控制调用其它程序组织单元的时候是 很有用的。

梯形图包含了一系列的网络，左右两边各有一个垂直的电流线，网络图仅限制于左右两母线之间的 范围内，在中间是由线圈触点和连接线组成的电路图。

每一个网络包含左边的一系列触点，这些触点根据布尔变量值的 TRUE 和 FALSE 来传递从左到右的开 和关的状态。每一个触点是一个布尔变量，如变量值为 TRUE，电路从左到右通过连接线就连通。否则右 边接收到“关“的值

下面是一个梯形图的例子，它由线圈和触点组成



触点

在梯形图中的每一个网络图的左边都有触点（触点是用两个平行线| |来表示），它用来表示电路的 “开” “关”状态。

这些状态与布尔变量 TRUE 和 FALSE 相一致。布尔变量属于每一个触点。如果变量值为 TRUE，那么 状态可以通过连接线从左边传到右边。否则，右边接收到的是“断开”。

触点可以并联使用，其中的一个并联分支必须传递“开”状态时，并联分支才能传递“开”。或者 触点串联连接，此时，触点必须传递“开”状态时，最后的触点才传递“开”，这些与串并联连接的电 路一致。

线圈

在梯形图网络图的右边有一些所谓的线圈，它们用（）表示并且只能通过水平线来连接。线圈传递 从左到右的连接状态，并且复制状态到布尔变量中，可以描述入口线的状态为“开”（对应布尔变量的 TRUE）或者“关闭”状态（对应布尔变量的 FALSE）。

触点和线圈也可以取否定值（在上例中的触头 SWITCH1 和线圈%QX3.0 是取否定值）。如果触点取否 定值（在触头符号中用“/”来表示），然后把它复制否定后的值到相应的布尔型变量中。如果一个触点 取否定值，仅且相应的布尔变量取到 FALSE 时，电路才能连通。

梯形图中的功能模块

可以在网络图中添加功能模块和程序，但它们必须具有布尔型值的输入和输出并且可以象触点那样 用在梯形图的左边。

梯形图中的功能块

在 接点和 线圈上你也可以在网络图中输入 功能块和程序， 它们必须有布尔变量的输入和输出并且 可以被用在和触点一样的地方，它在LD网络图左边。

置位/复位线圈

线圈也可以被定义设置或重置线圈。以“S”做为线圈被设置的标志：(S))。它从不在布尔变量上写 入TRUE。也就是说，如果一个变量被设为TRUE，它便被保留了下来。

以“R”做为线圈重置的标志：(R))。它从不在布尔变量上写入 FALSE：如果一个变量被设为 FALSE， 它便被保留了下来。

LD和FBD

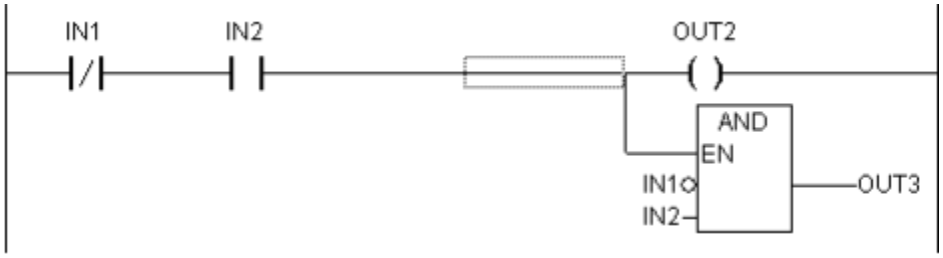
当应用LD是你很可能想用为了连接其他POUs的接触开关。令一方面，你可以用线圈把结果插入一个 可以在其他地方使用的全局变量你还可以在你的LD网络图中插入可执行命令。为此你导入一个POU和EN 输入。

这样的POUs是完全正常的有附加输入并以EN为标志的操作数、功能、程序或者 功能块。EN输入总是 BOOL类型并且它的意思是：当EN存在真值时，EN输入的POU求值。

EN POU 平行连接到线圈上。EN 输入连接到接点和线圈中间的连接线。输入 ON 信息在线上传送，这 个 POU 将被求值。

从 EN POU 开始，你可以创造和 FBD 相似的网络图

EN POU的LD 网络图的例子



5.7 操作块 Operator

- 1、IEC Operators（运算符）：

  ADD （加法）

  MUL （乘）

  SUB （减）

  DIV （除）

  MOD （求余）

- 2、Bitstring Operators（逻辑操作）：

  AND （与）

  OR （或）

  XOR （异或）

  NOT （非）

- 3、Bit-shift Operators（移位操作）：

  SHL （左移）

  SHR （右移）

  ROL （循环左移）

  ROR （循环右移）

- 4、Comparison Operators（比较运算）：

  GT （大于）

LT （小于）

LE （小于等于）

GE （大于等于）

EQ （等于）

NE （不等于）

- 5、Address Operators（地址）： ADR
- 6、Calling Operators（调用操作） : CAL
- 7、Type Conversion Functions（转型功能）：
  - BOOL_～ （布尔值转型）（ INT/STRING/TIME/～D/DATE/DT 等）
  - ～ _BOOL （转型成布尔值）（BYTE/INT/TIME/STRING 等）
  - INT_～_SINT/REAL （整数类型转换）
  - REAL_～/LREAL_～ （实数型/长实数型转型）（ INT 等）
  - TIME_～/TIME_OF_DAY（时间转型）（ STRING/DWORD/SINT 等）
  - DATE_～/DT_～ （日期转型）（ BOOL/INT/BYTE/STRING 等）
  - STRING_～ （字符串转型）（ BOOL/WORD/TIME 等） TRUNC（取整）
- 8、Numeric Functions（数据计算功能）：

ABS （取绝对值）

SQRT （开方）

LN （取自然对数）

LOG （取对数）

EXP （e 求幂）

SIN （正弦）

COS （余弦）

TAN （正切）

ASIN （反正弦）

ACOS （反余弦）

ATAN （反正切）

EXPT （求幂）

5.8 库文件 Library

5.8.1 Standard.lib 标准库

- 1.String function（字符串运算）：

  LEN （长度计算）

  LEFT （左取位）

  RIGHT （右取位）

  MID （中间取位）

  CONCAT （字符串叠加）

  INSERT （插入）

  DELETE （删除）

  REPLACE （代替）

  FIND （查找）

- 2. Trigger（触发保持）：

  R_TRIG （上升沿保持）

  F_TRIG （下降沿保持）

- 3. Counter（计数器）：

  CTU （上升沿计数）

  CTD （下降沿计数）

  CTUD （上升沿、下降沿计数）

- 1.4. Timer（计时器）：

  TP （触发计时器）

  TON （高电平计时器）

  TOF （低电平计时器）

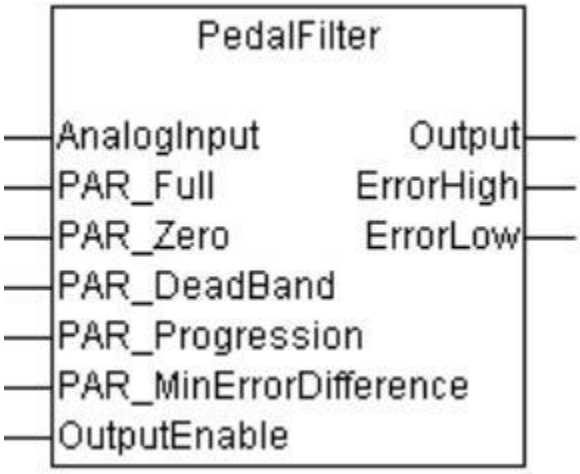
  RTC （运行时钟计时器）

5.8.2 AnalogFilter 模拟量标定库

功能描述：对模拟量输入信号进行标定，分为单向及双向标定。

1.单向标定模块

单向标定模块用于把输入模拟量进行预处理，把输出量范围调整到 0～65535。



参数说明：

- 1.AnalogInput 输入量 数据类型：UINT来自于传感器
- 2.OutPutEnable 使能开关

  数据类型：BOOL

  参数值为 TRUE 时，模块按当前值输出；为FALSE 时，保持上一次输出值。
- 3.PAR_Full

  数据类型：UINT

  取值范围： 0～255

  此参数用于调整最大输出值，当输入最大时，使输出接近 65535，小于 65535。
- 5. PAR_Zero

  数据类型：INT

  取值范围： 0～255

  此参数用于调整零位，使输出为零。
- 5 .PAR_DeadBand

  数据类型：UINT

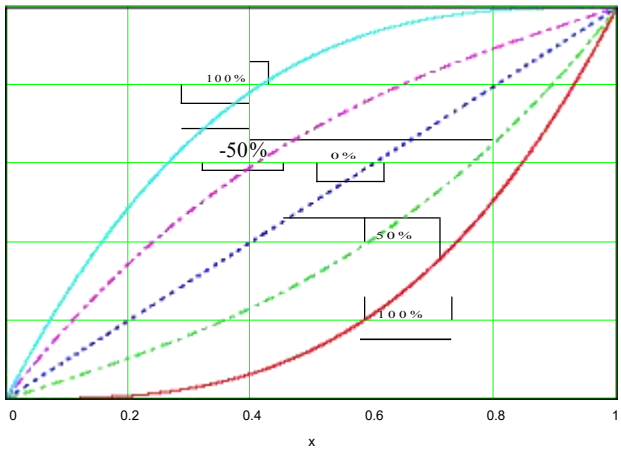
  取值范围： 0～255

  设置死区，当输入进入参数设置范围，输出为零。 6）PAR_Progression

  数据类型：SINT

  取值范围： -100～ 100

  设置输出缓冲曲线度，如图：



2.双向标定模块

双向标定模块用于把输入模拟量进行预

- 7. PAR_MinErrorDifference

  数据类型：UINT

  取值范围： 0～ 100

  设置出错范围，当输入超出范围时，激活 ErrorHigh 或ErrorLow 开关，并使输出为零。

- 8.Output 输出

  数据类型：UINT

  输出值范围为 0～65535

- 9. ErrorHigh 高位出错开关

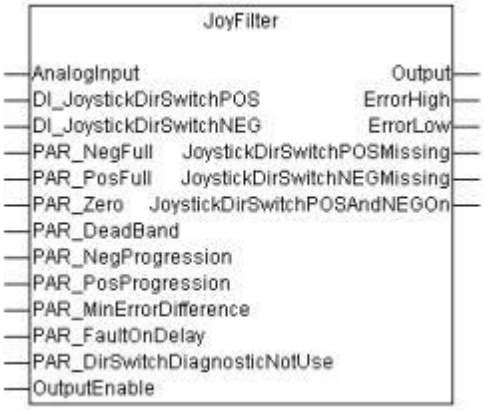
  数据类型：BOOL

  输入超出 PAR_Full + PAR_MinErrorDifference 范围， ErrorHigh 为 TRUE，并把输出置零。

- 10.ErrorLow 低位出错开关

  数据类型：BOOL

  输入超出 PAR_Zero－PAR_MinErrorDifference 范围， ErrorLow 为 TRUE，并把输出置零。 处理，把输出量范围调整到-32767～32767。



参数说明：

- 1. AnalogInput（同上）
- 2. DI_JoystickDirSwitchPOS 正向微动开关

  数据类型：BOOL
- 3 . DI_JoystickDirSwitchNEG 负向微动开关

  数据类型：BOOL
- 4.OutputEnable（同上）
- 5.PAR_NegFull

  数据类型：UINT

  取值范围： 0～255

  此参数用于调整负向最大输出值，使输出接近-32767，大于-32767。
- 6.PAR_PosFull

  数据类型：UINT

  取值范围： 0～255

  此参数用于调整正向最大输出值，使输出接近 32767，小于 32767。
- 7. PAR_Zero

  数据类型：INT

  取值范围： 0～255

  调整零位，当传感器在中位时，使输出为零。
- 8 .PAR_DeadBand（同 2.1 ）
- 9. PAR_NegProgression（同 2.1 ）

- 10.PAR_PosProgression（同 2.1 ）
- 11.PAR_MinErrorDifference（同 2.1 ）
- 12.PAR_Faul～nDelay 过载延时

  数据类型：UINT

  取值范围： 0～255
- 13.PAR_DirSwitchDiagnosticNotUse 数据类型：BOOL

  此参数为 TRUE 时，微动开关无效，为FALSE 时，微动开关有效 14） Output 输出

  数据类型：INT

  输出数值范围-32767～32767
- 15.ErrorHigh（同上）

  数据类型：BOOL

  当输入超出 PAR_Full +PAR_MinErrorDifference 范围时， ErrorHigh 为 TRUE，并把输出 置零。
- 16）ErrorLow（同上）

  数据类型：BOOL

  当输入超出 PAR_Zero－PAR_MinErrorDifference 范围时， ErrorHigh 为 TRUE，并把输出 置零。
- 17.JoystickDirSwitchPOSMissing

  数据类型：BOOL

  正向微动开关出错时，值为 TRUE。
- 18.JoystickDirSwitchNEGMissing

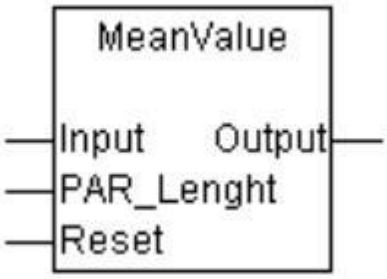
  数据类型：BOOL

  负向微动开关出错时，值为 TRUE。
- 19.JoystickDirSwitchPOSAndNEGOn

  数据类型：BOOL

  正、负向微动开关同时为 TRUE 时，值为 TRUE。

3. 平均值模块



参数说明:

- 1 .Input

  数据类型: UINT

- 2.Reset

  数据类型: BOOL

  值为 TRUE 时不进行平均值运算。

- 3 .PAR_Length

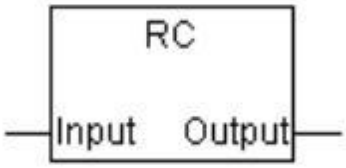
  数据类型: UINT

  取数长度，如值为 5，则每 5 个值求一次平均值，并输到 output。

- 4.Output

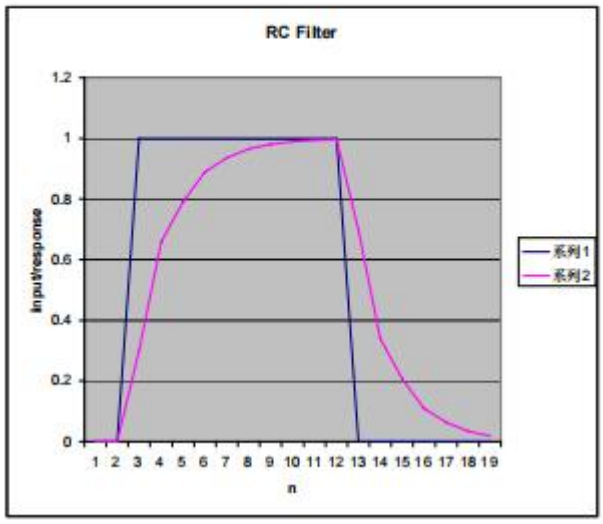
  数据类型: UINT

4. RC 滤波模块



滤波公式如下：

$$y(z) := x(z)\cdot 0.6 + y\left(z^{-1}\right)\cdot 0.4$$



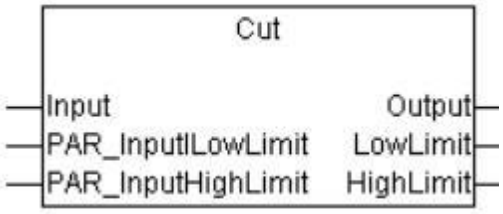
- 1. Input

  数据类型：UINT

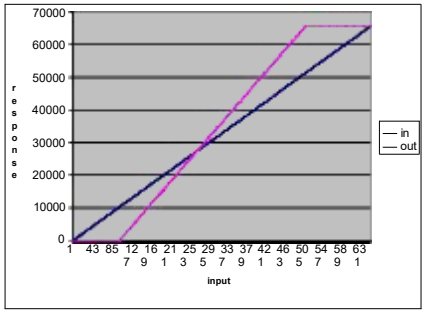
- 2. Output

  数据类型：UINT

5. Cut 模块



模块的功能是把输入值进行预处理，调整斜率，消除两端非线性影响。



- 1. Input

  数据类型：UINT

- 2.PAR_InputLowLimit

  数据类型：UINT

  当输入值小于 256*PAR_InputLowLimit)时，输出为零。

- 3 .PAR_InputHighLimit

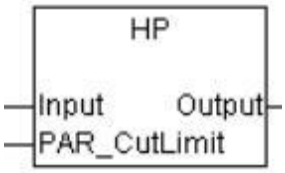
  数据类型：UINT

  当输入小于 65535 时，设置参数 PAR_InputHighLimit=256/INPUT。

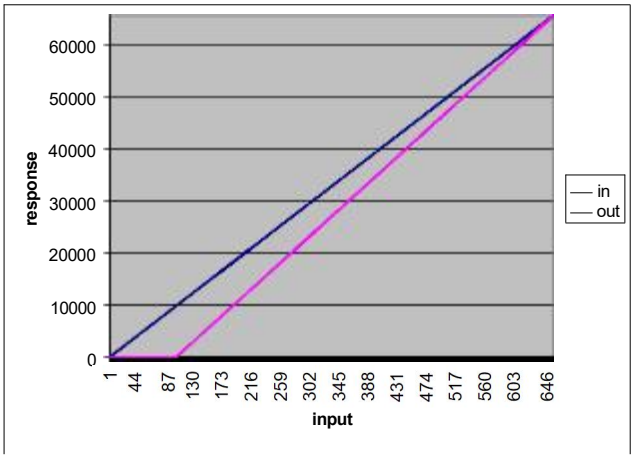
- 4.Output

  数据类型：UINT

6. HP 模块



HP 模块用于设定死区。



Out=(65535/(65535-655* PAR_CutLimit))*(INPUT-655* PAR_CutLimit)。

- 1.Input

  数据类型：UINT

- 2.PAR_CutLimit 调整参数

  数据类型：UINT

- 3 .Output

  数据类型：UINT

7. Transfer 模块

Transfer 模块用于设置缓冲曲线度，当 Zero为 TRUE 时,输出为零，当 Full 为 TRUE, 而 Zero 不为零时，输出为 65535。

缓冲曲线度公式: Y2 表正向，Y3 表负向。

而 Zero 不为零时，输出为 65535。

缓冲曲线度公式: Y2 表正向，Y3 表负向。

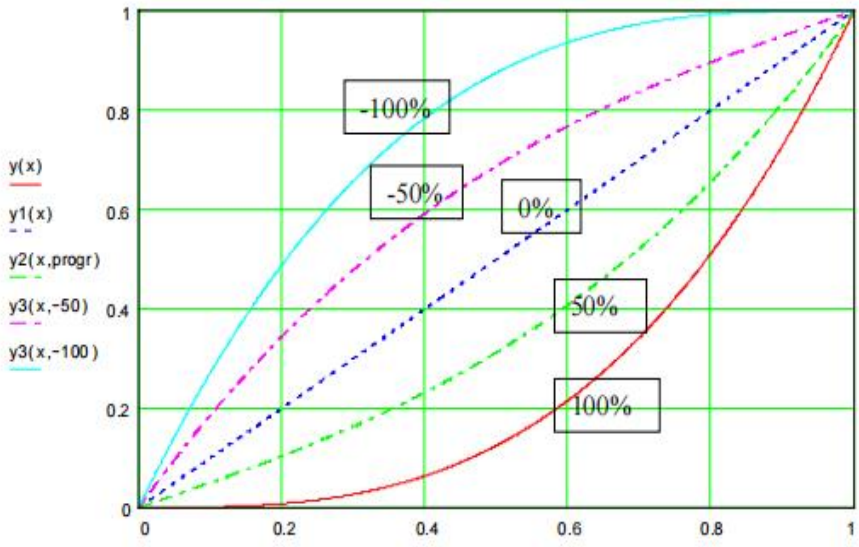
$x := 0, 0.001 .. 1 \qquad progr := 50$

$y(x) := x^3$

$y1(x) := x$

$$y2(x, progr) := y(x)\cdot\frac{progr}{100} + \frac{y1(x)\cdot(100 - progr)}{100}$$

$$y3(x, progr) := 1 - \left[y(1 - x)\cdot\frac{-progr}{100} + \frac{y1(1 - x)\cdot(100 + progr)}{100}\right]$$



- 1 .Input

  数据类型: UINT

- 2.Zero

  数据类型: BOOL

  当 Zero 为 TRUE 时,输出为零。

- 3 .Full

  数据类型: BOOL

  当 Full 为 TRUE,而 Zero 不为零时，输出为 65535。

- 4.OutputEnable

  数据类型: BOOL

  当此参数值为 TRUE 时，模块按预定算法执行；为FALSE 时，保持上一次输出。

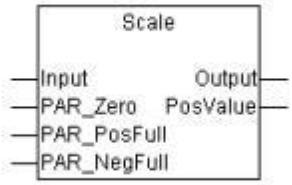
- 5 .PAR_Progression 曲线度设置参数

  数据类型: INT

  6.Output

  数据类型: UINT 2.

8. Scale 模块



- 1.Input

  数据类型: UINT

- 2.PAR_Zero

  数据类型: INT

  设置零位，当输入为 PAR_Zero * 256 时,输出为 0.

- 3 .PAR_PosFull

  数据类型: UINT

当 input 值>=PAR_PosFull * 256 时， 输出为 65535 ， PosValue 为 TRUE。

- 4. PAR_NegFull

  数据类型：UINT

  当 input 值<= PAR_NegFull * 256, 输出为-65535 ,PosValue 为 FALSE。

- 5.Output

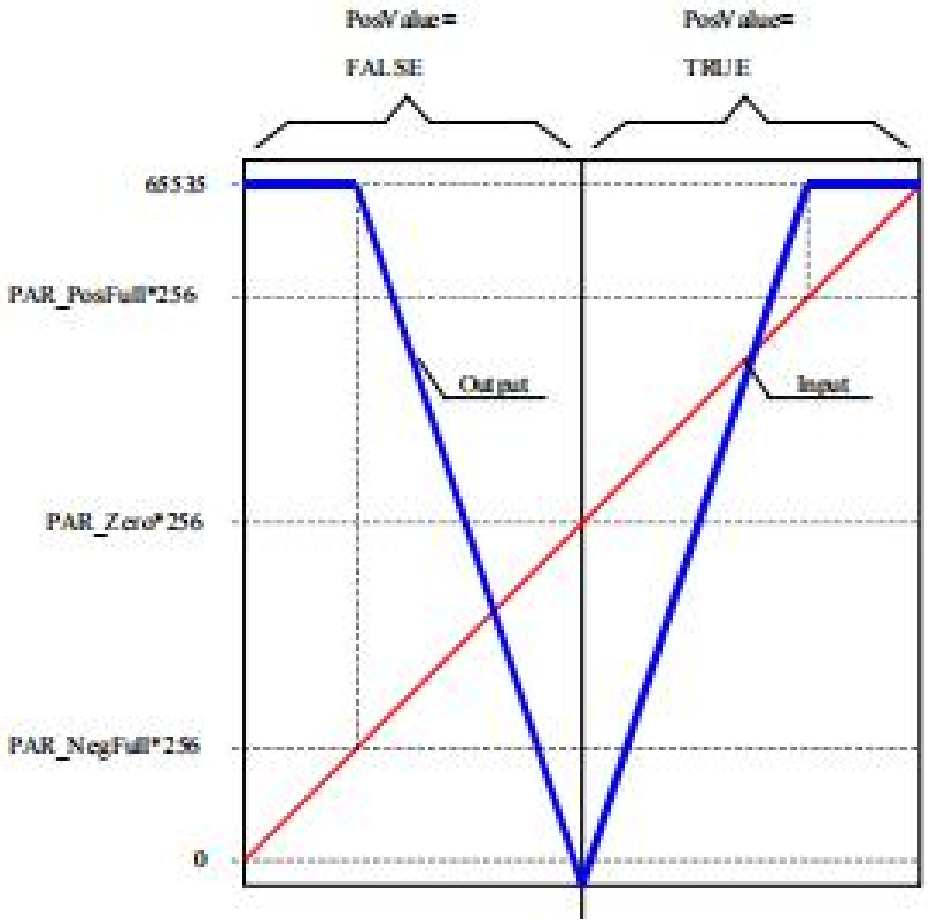
  数据类型：UINT

- 6.PosValue

  数据类型：BOOL

  PAR_Zero * 256 > Input 值，值为 TRUE； PAR_Zero * 256 < Input 值，值为 FALSE。

  Scale 用于把输入为 0～65535 的值调整到-32767～32767.如下图：



5.8.3 调试、联机功能

采样追踪

采样追踪允许你追踪变量的连续变化的值，它依赖于所谓的触发事件，触发事件是先前定义的布尔 变量（触发变量）的上升沿或下降沿。CoDeSys 允许对 20 个变量进行追踪，每一个变量可以追踪 500 个 值。

调试

CoDeSys 的调试功能可以让你很容易的找到错误。

为了调试，运行‘工程’‘选项’命令并且在 生成选项对话框中选择动态调试。

断点

断点是程序处理过程中停止的位置，因而它可以在程序中的特定位置观察变量值的变化。

断点可以在编辑器中设置，在文本编辑器中断点在行的编号处设置,在连续功能图中和梯形图中是在 网络编号处设置,在 CFC 中是在程序组织单元处设置,在 SFC 中是在步处设置，在功能模块图的实例中不 能设置断点。

注意: CoDeSys SP 全 32 位运行系统只要程序在断点处停止运行，它将取消与之相连任务的看门狗功能.

单步

单步是:

在指令表中:执行程序直到运行 CAL LD 和 JMP 命令。 在结构化文本中：执行下一条指令。

在功能模块图 梯形图中： 执行下一步网络。

在顺序功能图中：继续目前的 动作 直到下一个 步开始。 通过一步一步的运行你可以检查程序中的逻辑错误。

单循环

如果选择了单循环，每一个循环结束，执行也就结束。

联机模式下改变值

在操作过程中，变量可以设置为一个特定的值（写入新值）或者在每一个循环之后重新定义为特定 的值（强制新值）。在联机模式下可以通过双击变量的值来改变它的值，布尔变量从TRUE变为FALSE或从 FALSE变为TRUE。对于每一种类型的变量都可以打开写入变量对话框。在这里可以编辑变量的真实值。

监视

在联机模式下，所有的显示变量从控制器中读出并及时的显示。你可以在定义和程序编辑器中找到 这些显示。你也可以在观察和接收器中读出变量的当前值并且可以看到它们。如果要监视功能模块的实 例中的变量,相应的实例块必须已经打开。在监视 VAR_IN_OUT 变量时，不引用的值将输出。

在监视指针时，指针和不引用的值都将在声明部分输出。在程序部分，只有指针输出： + --pointervar = '<'pointervalue'>'

在不引用值中的 POINTER 也相应的显示。在行上双击或在交叉上单击，显示或是展开或是收缩。 在执行部分，显示指针的值。对于不引用，将显示不引用的值。

监视数组元素：数组元素除了由常量指出的之外，还有由变量指出的：

anarray[1] = 5

anarray[i] = 1

如果索引中包含有表达式(例如， [i+j] or [i+1])，元素不能显示出来。

请注意：如果已经达到了被监视变量的最大编号，对于随后的变量不是显示当前的值，而是显示字 符串“监视的变量太多”。

仿真

在模拟过程中，创建的 PLC 程序不在实际的 PLC 中运行，而是 CoDeSys 系统中的计算器中运行。所 有的联机功能都是可用的。它允许你在无需 PLC 硬件的情况下检测逻辑的正确性。

注意：外部库文件的 POU 是不能运行在模拟的模式的。

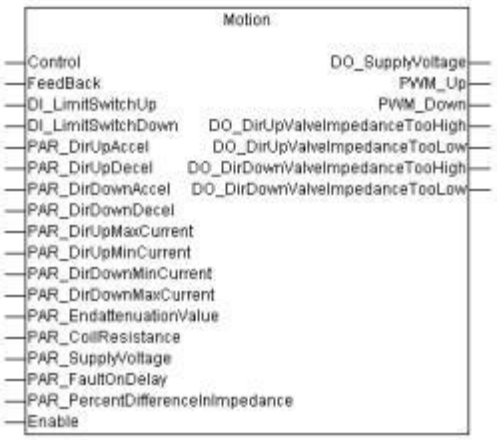
日志

日志记录着用户的操作、内部进程、状态变换和联机模式处理过程中发生的意外的情形。它用来监 视和跟踪错误。

5.8.4 PWMControlvoltageAmpere -funktio 库

1. Motion 模块

Motion 模块主要用于控制比例阀， 有两路 PWM 输出，控制信号范围-32767 ~ 32767，当控制信号在 0 ～32767， PWM Up 输出，值 0～32767；当控制信号在 0 ～-32767， PWM down 输出，值 0～-32767。由参数控制最大最小输出电流范围。



- 1 .Control 控制信号

  数据类型: INT

  取值范围: -32767 ～32767。

- 2. Feedback

  数据类型: UINT;

  反馈电流信号;

  取值范围: 0 ～65535。

- 3 .DI_LimitSwitchUp

  数据类型: BOOL;

  正向限位开关，为 TRUE 时模块输入为 PAR_EndattenuationValue*Control。

- 4.DI_LimitSwitchDown

  数据类型: BOOL;

  逆向限位开关，为 TRUE 时模块输入为 PAR_EndattenuationValue*Control。

- 5 .Enable

  数据类型: BOOL;

  使能开关，为 FALSE 时， DO_SupplyVoltage 为 FALSE，输出为零。

- 6.PAR_DirUpAccel

数据类型: USINT;

正向缓冲加速参数;

取值范围:1 ～255。

- 7.PAR_DirUpDecel

  数据类型: USINT

  正向缓冲减速参数;

  取值范围:1 ～255。

- 8 .PAR_DirDownAccel

  数据类型: USINT

  逆向缓冲加速参数;

  取值范围:1 ～255。

- 9.PAR_DirDownDecel

  数据类型: USINT

  逆向缓冲减速参数;

  取值范围:1 ～255。

- 10.PAR_DirUpMaxCurrent

  数据类型: UINT

  正向最大电流;

  取值范围: 0 ～ 1000 mA。

- 11 .PAR_DirUpMinCurrent

  正向最小电流;

  取值范围: 0 ～ 1000 mA。

- 12.PAR_DirDownMinCurrent

  数据类型: UINT

  逆向最小电流;

  取值范围: 0 ～ 1000 mA。

- 13 .PAR_DirDownMaxCurrent

  数据类型: UINT

  逆向最大电流;

取值范围: 0 ～ 1000 mA。

- 14.PAR_EndattenuationValue

  数据类型: UINT

  限位参数;

  取值范围: 0 ～ 100 %

- 15 .PAR_CoilResistance

  数据类型: USINT

  比例阀阻抗;

  取值范围: 0 ～255 Ω

- 16.PAR_SupplyVoltage

  数据类型: USINT

  供电电压;

  取值范围: 0 ～255 V

- 17.PAR_Faul～nDelay

  数据类型: UINT

  出现错误延迟时间;

  取值范围: 0 ～ 65535 ms

- 18.PAR_PercentDifferenceInImpedance

  数据类型: USINT

  取值范围: 0 ～ 100 %

- 19.DO_SupplyVoltage

  数据类型: BOOL

  当 Enable 为 TRUE 且 control <>0 时，为 TRUE。

- 20.PWM_Up

  数据类型: UINT 正向输出;

  取值范围: 0 ～ 32767

- 21 .PWM_Down

  数据类型: UINT 逆向输出;

  取值范围: 0 ～ 32767

- 22.DO_DirUpValveImpedanceTooHigh

  数据类型：BOOL

  正向时，当 impedance 高于：（PAR_PercentDifferenceInImpedance*Impedance./100， 且延时到 PAR_Faul～nDelay ，该值为 TRUE。

- 23 .DO_DirUpValveImpedanceTooLow

  数据类型：BOOL

  正向时，当 impedance 低于：(PAR_PercentDifferenceInImpedance*Impedance)/100， 且延时到 PAR_Faul～nDelay ，该值为 TRUE。

- 24.DO_DirDownValveImpedanceTooHigh

  数据类型：BOOL

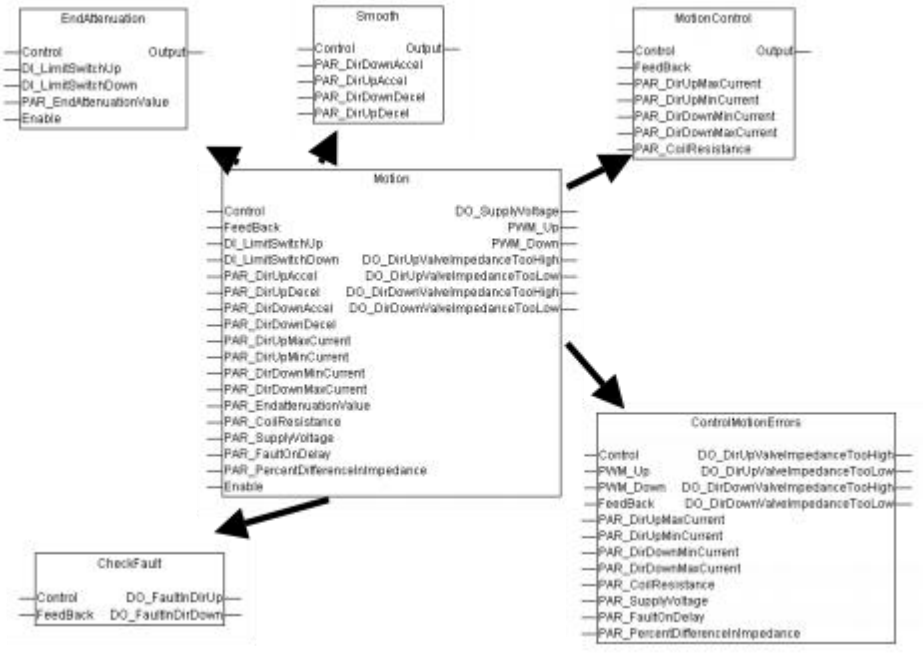
  逆向时，当 impedance 高于（PAR_PercentDifferenceInImpedance*Impedance）/100， 且延时到 PAR_Faul～nDelay ，该值为 TRUE。

- 25 . DO_DirDownValveImpedanceTooLow

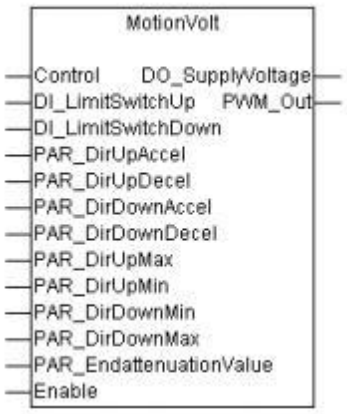
  数据类型：BOOL

  逆向时，当 impedance 低于:(PAR_PercentDifferenceInImpedance*Impedance)/100，且延时到 PAR_Faul～nDelay ，该值为 TRUE。

Motion 块有以下子块：



2. Motion Volt 模块



该模块也用于控制比例阀，输出 PWM 信号，控制信号来自 joystickfilter 模块(-32767 ~ 32767)。 输出为 0 ～ 32767。

- 1 . Control

  数据类型: INT 控制信号;

  取值范围: -32767 ～ 32767

- DI_LimitSwitchUp

  数据类型: BOOL 参见 3.1。

- 3 . DI_LimitSwitchDown

  数据类型: BOOL 参见 3.1。

- 4. Enable

  数据类型: BOOL 参见 3.1。

- 5 . PAR_DirUpAccel

  数据类型: USINT 参见 3.1;

  取值范围: 1 ～ 255。

- 6.PAR_DirUpDecel

  数据类型: USINT 参见 3.1;

  取值范围: 1 ～ 255

- 7.PAR_DirDownAccel

  数据类型: USINT

  参见 3.1;

  取值范围: 1 ～ 255

- 8 . PAR_DirDownDecel

  数据类型: USINT 参见 3.1;

  取值范围: 1 ～ 255

- 9.PAR_DirUpMax

  数据类型: UINT

  正向输出最大值为 PWM 的百分比。例， 80 %.

  取值范围: 0 ～ 100 % 10）

- 10.PAR_DirUpMin

  数据类型: UINT

  正向输出最小值为 PWM 的百分比。例， 20 %.

  取值范围: 0 ～ 100 % 11 ）

- 11.PAR_DirDownMin

  数据类型: UINT

  逆向输出最大值为 PWM 的百分比。例， 80 %.

  取值范围: 0 ～ 100 % 12）

- 12.PAR_DirDownMax

  数据类型: UINT

  逆向输出最小值为 PWM 的百分比。例， 20 %.

  取值范围: 0 ～ 100 %

- 13 .PAR_EndattenuationValue

  数据类型: UINT

  参见 3.1;

  取值范围: 0 ～ 100 %

- 14.DO_SupplyVoltage

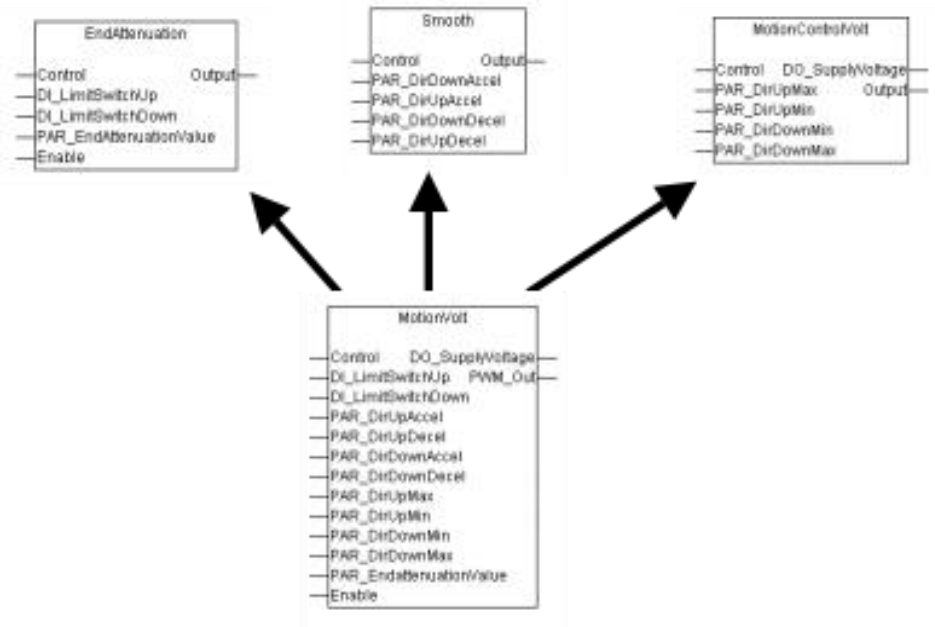
  数据类型: BOOL 参见 3.1;

- 15 .PWM_Out

  数据类型: UINT PWM 输出;

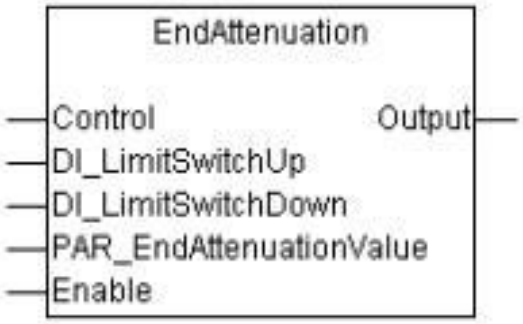
  取值范围: 0 ～ 32767

Motion Volt 块有以下子块：



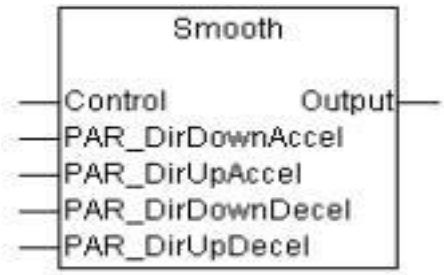
3.3. EndAttenuation 模块

参数说明参见 3.1；



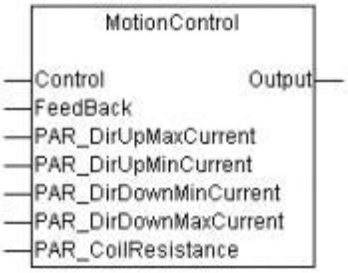
3.4. Smooth 模块

参数说明参见 3.1；



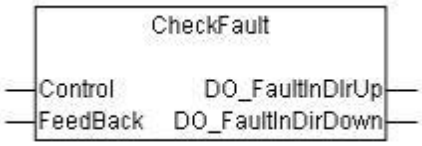
3.5. MotionControl 模块

参数说明参见 3.1;



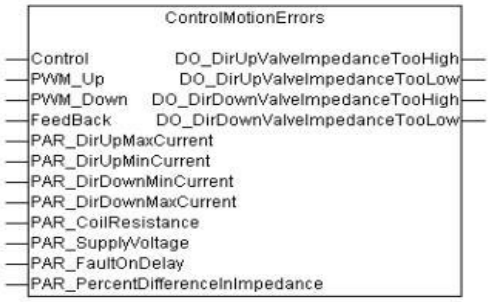
3.6. CheckFault 模块

参数说明参见 3.1;



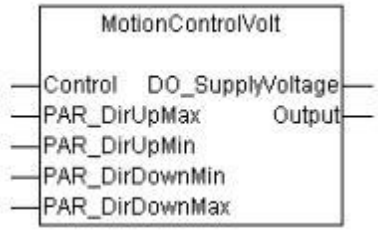
3.7. ContolMotionErrors 模块

参数说明参见 3.1;



3.8. MotionControlVol模块

参数说明参见 3.2;



6 硬件功能库的说明

在 HW 库里，包含一些与硬件功能有关的模块，可通过这些模块对一些参数进行设置。例如： 设置 PWM 信号的频率等。

6.1 Configure_PI 功能模块

Configure_PI 模块用于脉冲计数，它有两个输入通道 A、B。模块如下：

当要进行单路计数时，设置 B=16#10，A 通道接脉冲信号输入通道号 n（n 为输入通道地址减 150），对应的脉冲计数值在 对应的%IWd（d 为 A 输入通道地址加 10），通道号按下表进行设置。

	Channel #		**Pulse count IW**	
IW	**2023**	**2024**	**2023**	**2024**
130	0		120	
131	1		121	
132	2		122	
133	3		123	
134	4		124	
135	5		125	
136	6		126	
137	7		127	
150	8	0	160	160
151	9	1	161	161
152	10	2	162	162
153	11	3	163	163
154	12	4	164	164
155	13	5	165	165
156	14	6	166	166
157	15	7	167	

当有两路脉冲输入，而两脉冲信号的相位差为 90 度时，可进行增量式计数，A、B 分别接两

路脉冲的通道号，计数值在对应的输入存储器里%Iwd(d 为A 通道对应的输入地址加 10)。 Configure PI 的输入：

- 1.A

  数据类型: UINT

  功能描述: 连接到脉冲输入信号 A.

  数据范围: 0 to 65535

- 2.B

  数据类型: UINT

  功能描述: 连接到脉冲输入信号 B.

  数据范围: 0 to 65535 Configure PI

- 3 .Memory outputs

  数据类型: UINT

  描述:模块输出在输出存储器的 %IW 160 - 167.

  数据范围: 0 to 65535

6.2 Reset PI 功能模块

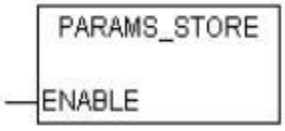
Reset PI 功能模块如下图，功能是把脉冲输入计数器置零，软连接口 CH 接所要复位的通道 号。

如果要用 B 通道脉冲上升沿复位 A 通道的脉冲计数值（或用 A 脉冲的上升沿复位 B 通道的 计数值），则 CH 的数值如下公式所示：

CH = (*Resetting channel number*+1) * 16 + *Channel to reset*

6.3 Params Store 功能模块

Params strore 模块的功能是把控制器的参数值（%MWn 的设定值）闪存到控制器的相应存储 区。当 ENABLE 为TRUE 时闪存。



6.4 Set PWM frequency 功能模块

Set PWM frequency 模块用于设置输出 PWM 信号的频率.可设频率范围从 40 到 2550 Hz。

可设置输出通道为%QW100 到%QW111 和%QW116 到%QW119。FREQ 设定值为 (实际频率)/10。

Output Word	Channel
100 to 103	0 to 3
104 to 107	4 to 7
108 to 111	8 to 11
116 to 119	16 to 19
120 to 123	20 to 23
Output Word	Channel
112	12
113	13
114	14
115	15

Set PWM frequency 的输入

- 1.Freq

  数据类型：USINT

  描述：(实际频率)/10 数值范围：4 to 255

- 2.Ch

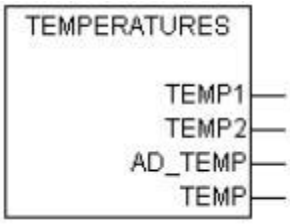
  数据类型：USINT

  描述：Connect the channel as the Sheet 2 and 3 shows.

  数值范围：0 to 23℃

6.5 Temperatures 功能模块

Temperature 功能模块用于读取控制器内部温度，



Temperature 功能模块有两路报警输出 (TEMP 1 and TEMP 2)；一路模拟量输出 (AD_TEMP) ，可通过下述公式计算温度值。还有一路实际的温度值输出(TEMP) 。

bit_per_deg = 81,263 bit_offset = 5,177 * 10^3

temperature = (AD_TEMP - bit_offset) / bit_per_deg

Examble:

AD_TEMP = 8470

tempereture = (8470 - 5,177 * 10^3) / 81,263 = 40,519 。C

Temperatures 输出

- 1 .Temp1

  数据类型：BOOL

  功能描述:当温度超过 74 ℃ 时为 TRUE。

- 2.Temp2

  数据类型：BOOL

  功能描述:当温度低于 -40 ℃ 为 TRUE。

- 3 .AD_Temp

  数据类型：UINT

  功能描述： 模拟信号输出，可通过上面公式计算实际温度值； 数值范围：- 40 to + 125 ℃

- 4.Temp

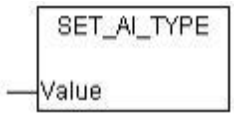
  数据类型：SINT

  功能描述： 输出的实际温度值

  数值范围：- 40 to + 125 ℃

6.6 SET_AI_TYPE 模块

当模块的模拟量输入有电流输入时，需进行设置，通过设置模块的 Value 值，可设定不同的通道为电流输入。



Value 设置不同的数值，可设置相对应的针脚为电流输入（系统默认为电压输入）。

PLC Set 1 的设置图表：

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
XM3.3		I		I		I		I		I		I		I		I		I		I		I		I		I		I		I		I
XM3.5			I	I			I	I			I	I			I	I			I	I			I	I			I	I			I	I
XM3.7					I	I	I	I					I	I	I	I					I	I	I	I					I	I	I	I
XM3.10					I	I	I	I					I	I	I	I					I	I	I	I					I	I	I	I

PLC Set 2 的设置图表

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
XM1.12		I		I		I		I		I		I		I		I		I		I		I		I		I		I		I		I
XM2.12			I	I			I	I			I	I			I	I			I	I			I	I			I	I			I	I
XM3.5					I	I	I	I					I	I	I	I					I	I	I	I					I	I	I	I
XM3.6					I	I	I	I					I	I	I	I					I	I	I	I					I	I	I	I
XM3.7					I	I	I	I					I	I	I	I					I	I	I	I					I	I	I	I
XM3.8									I	I	I	I	I	I	I	I									I	I	I	I	I	I	I	I
XM3.13																	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
XM3.14																	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I

7 CoDeSys 程序库

7.1 字符串功能

注意：字符串功能不具有'线程安全'：当使用任务时,字符串功能只能用于一个任务中,如果同一功能用于不同任务，有被覆盖的危险.

See also:

LEN

LEFT

RIGHT

MID

CONCAT INSERT DELETE REPLACE FIND

LEN

返回字符串长度。

输入 STR 便是类型 STRING，功能的返回值是 INT 类型。

- IL 例子:

```
LD 'SUSI'
LEN

T VarINT1 (* 结果是 4 *)
```

- FBD 例子:

- ST 例子:

```
VarSTRING1 := LEN ('SUSI');
```

注意：字符串功能并非"thread safe"：使用任务时，字符串公民只可用在单任务中。如果不同任务中使用相同的 功能，便有被覆盖的危险。

- **LEFT**

Left 返回，已知字符串为初始化字符串。

输入 STR 即类型 STRING，SIZE 为 INT 类型，此功能的返回值为 STRING 类型。LEFT(STR,SIZE)意思是：在字符串 STR 从右边取第一个 SIZE 字节。

➢ IL 例子:

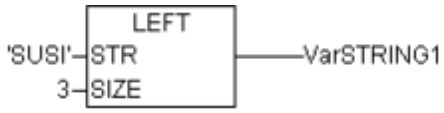
LD 'SUSI'

LEFT 3

ST VarSTRING1 (* 结果是 'SUS' *)

FBD 例子:

ST 例子:



VarSTRING1 := LEFT ('SUSI',3);

注意: 字符串功能并非"thread safe":使用任务时,字符串公民只可用在单任务中。如果不同任务中使用相同的 功能,便有被覆盖的危险。

- **RIGHT**

Right 返回,已知字符串为初始化字符串。

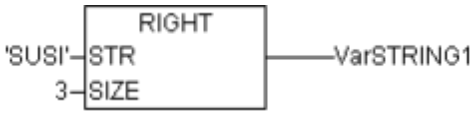
RIGHT(STR,SIZE)意思是:从字符串 STR 的右边取第一个 SIZE 字节。

- IL 例子:

LD SUSI' 3

VarSTRING1 (* 结果是 'USI' *)

- ST FBD 例子:



- ST 例子:

VarSTRING1 := RIGHT ('SUSI',3);

注意: 字符串功能并非"thread safe":使用任务时,字符串公民只可用在单任务中。如果不同任务中使用相同的 功能,便有被覆盖的危险。

- **MID**

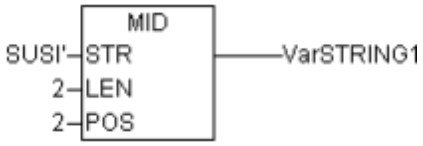
MID 从一个字符串中返回一部分字符串。

输入 STR 为类型 STRING,而 LEN 和 POS 为类型 INT,功能的返回值为类型 STRING

- IL 例子:

```
LD  'SUSI' MID 2,2
ST  VarSTRING1 (* 结果是 'US' *)
```

- FBD 例子:



- ST 例子:

```
VarSTRING1 := MID ('SUSI',2,2);
```

注意: 字符串功能并非“thread safe”：使用任务时，字符串公民只可用在单任务中。如果不同任务中使用相同的 功能，便有被覆盖的危险。

- **CONCAT**

两个字符串的连接(合并)

输入变量 STR1 和 STR2 以及功能的返回值都是类型 STRING。

➢ IL 例子:

```
LD          'SUSI'
CONCAT      'WILLI'
ST          VarSTRING1 (* 结果是 'SUSIWILLI' *)
```

FBD 例子:

- ST 例子:

```
VarSTRING1 := CONCAT ('SUSI','WILLI');
```

注意: 连接 CONTACT 功能如果嵌套三层以上便失灵。

注意: 字符串功能并非“thread safe”：使用任务时，字符串公民只可用在单任务中。如果不同任务中使用相同的功能，便有被覆盖的危险。

- **INSERT**

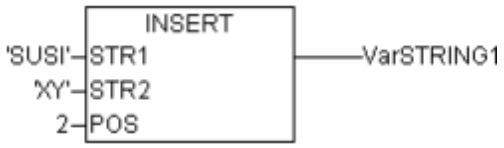
INSERT 可以在定义的点把一个字符串插入到另一个字符串中。

输入变量 STR1 和 STR2 为 STRING 类型，POS 是 INT 类型，功能返回值是 STRING 类型。 INSERT(STR1,STR2,POS)意思是：在 POS 之后将 STR2 插入到 STR1 中。

- IL 例子:

```
LD    'SUSI'  INSERT 'XY',2
ST    VarSTRING1 (* 结果是 'SUXYSI' *)
```

- FBD 例子:



- ST 例子:

```
VarSTRING1 := INSERT ('SUSI','XY',2);
```

注意：字符串功能并非“thread safe”：使用任务时，字符串公民只可用在单任务中。如果不同任务中使用相同的 功能，便有被覆盖的危险。

DELETE

DELETE 可以在定义的位置把一个较大的字符串的一部分删除掉。

输入变量 STR 为 STRING 类型，LEN 和 POS 为 INT 类型，功能的返回值是 STRING 类型。DELETE(STR,L,P)意思是：从以 P 位置开始的字节处将 L 字节删除。

- IL 例子:

```
LD    'SUXYSI' DELETE 2,3
ST    Var1 (* 结果是 'SUSI' *)
```

- FBD 例子:

- ST 例子:

```
Var1 := DELETE ('SUXYSI',2,3);
```

注意：字符串功能并非“thread safe”：使用任务时，字符串公民只可用在单任务中。如果不同任务中使用相同的 功能，便有被覆盖的危险。

REPLACE

REPLACE 可用第三个字符串从较大的字符串中替换掉部分字符串。输入变量 STR1 和 STR2 为 STRING 类型，LEN 和 POS 为 INT 类型。REPLACE(STR1,STR2,L,P)意思是：从以 P 位置开始的 STR2 把 STR1 从 L 字节处替换。

- IL 例子:

```
LD          'SUXYSI'
REPLACE     'K',2,2
ST          VarSTRING1 (* 结果是 'SKYSI' *)
```

- FBD 例子:

- ST 例子:

```
VarSTRING1 := REPLACE ('SUXYSI','K',2,2);
```

注意：字符串功能并非“thread safe”：使用任务时，字符串公民只可用在单任务中。如果不同任务中使用相同的 功能，便有被覆盖的危险。

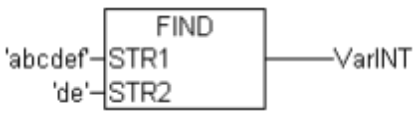
FIND

FIND 在一个字符串中查找到某部分字符串。输入变量 STR1 和 STR2 都是 STRING 类型，功能的返回值是 STRING 类型。FING(STR1,STR2)意思是：查找 STR1 第一次出现在 STR2 处时，后者第一个字节的位置。

- IL 例子:

```
LD 'abcdef'
INT 'de'
ST VarINT1 (* 结果是 '4' *)
```

- FBD 例子:



- ST 例子:

```
arINT1 := FIND ('abcdef','de');
```

注意：字符串功能并非“thread safe”：使用任务时，字符串公民只可用在单任务中。如果不同任务中使用相同的 功能，便有被覆盖的危险。

7.2 双稳功能程序

SR

重置双稳定功能模块

Q1 = RS (SET, RESET1) 的意思是: Q1 = NOT RESET1 AND (Q1 OR SET)

输入变量 SET 和 RESET1 以及输出变量 Q1 都是 BOOL 类型。

声明如下所示:RSInst : RS ;

- IL 例子:

```
CAL RSInst(SET:= VarBOOL1,RESET1:=VarBOOL2)
LD RSInst.Q1 ST VarBOOL3
```

- FBD 例子:

- ST 例子:

```
RSInst(SET:= VarBOOL1 , RESET1:=VarBOOL2 ); VarBOOL3 := RSInst.Q1 ;
```

RS

重置双稳定功能模块

Q1 = RS (SET, RESET1) 的意思是: Q1 = NOT RESET1 AND (Q1 OR SET)

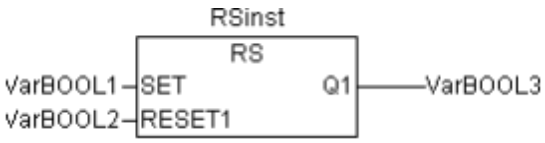
输入变量 SET 和 RESET1 以及输出变量 Q1 都是 BOOL 类型。

声明如下所示:RSInst : RS ;

- IL 例子:

```
CAL RSInst(SET:= VarBOOL1,RESET1:=VarBOOL2) LD RSInst.Q1
ST VarBOOL3
```

- FBD 例子:



- ST 例子:

```
RSInst(SET:= VarBOOL1 , RESET1:=VarBOOL2 ); VarBOOL3 := RSInst.Q1 ;
```

SEMA

软件信号量(可断续的)

BUSY = SEMA(CLAIM, RELEASE) 的意思是: BUSY := X;

IF CLAIM THEN X:=TRUE;

ELSE IF RELEASE THEN BUSY := FALSE; X:= FALSE; END_IF

X 为内部布尔变量，初始化时是 FALSE。输入变量 CLAIM 和 RELEASE 以及输出变量 BUSY 都是 BOOL 变量。

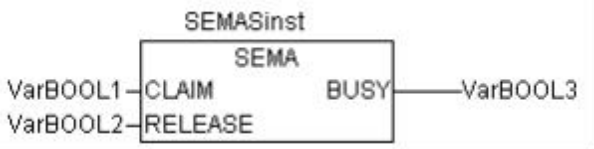
如果调用 SEMA 时，BUSY 为 TURE，即 SEMA 已经被分配了值(SEMA 被调用时，CLAIM＝TRUE)。若 BUSY 是 FALSE，不调用 SEMA 或已被启动(RELEASE=TRUE)。

声明如下所示:SEMAInst : SEMA ;

- IL 例子:

```
CAL SEMAInst(CLAIM:=VarBOOL1,RELEASE:=VarBOOL2
LD SEMAInst.BUSY
ST VarBOOL3
```

- FBD 例子:



- ST 例子:

```
SEMAInst(CLAIM:= VarBOOL1 , RELEASE:=VarBOOL2 );
VarBOOL3 := SEMAInst.BUSY;
```

7.3 触发器

R_TRIG

R_TRIG 功能块触发一个上升的边界。 FUNCTION_BLOCK R_TRIG

```
VAR_INPUT
CLK : BOOL; END_VAR
VAR_OUTPUT
Q : BOOL; END_VAR
VAR
M : BOOL := FALSE; END_VAR
Q := CLK AND NOT M; M := CLK;
```

只要输入变量 CLK 为 FALSE，输出 Q 和帮助变量 M 依然是 FALSE。一旦 CLK 返回 TRUE，Q 首先返回 TRUE，那么 M 就会被设置为 TRUE。此意味着，没调用一次此功能，Q 便会返回 FALSE，直到 CLK 边界已 下降，随即边界再次上升。

声明如下所示:RTRIGInst : R_TRIG ;

- IL 例子:

```
CAL RTRIGInst(CLK := VarBOOL1) LD  RTRIGInst.Q
ST  VarBOOL2
```

- FBD 例子:

- ST 例子:

```
RTRIGInst(CLK:= VarBOOL1); VarBOOL2 := RTRIGInst.Q;
```

F_TRIG

F_TRIG 功能块触发一个下降的边界。 FUNCTION_BLOCK F_TRIG

```
VAR_INPUT
CLK: BOOL; END_VAR
VAR_OUTPUT
Q: BOOL; END_VAR
VAR
M: BOOL := FALSE; END_VAR
Q := NOT CLK AND NOT M; M := NOT CLK;
```

只要输入变量 CLK 返回 TRUE，输出 Q 和帮助变量 M 依然是 FALSE。一旦 CLK 返回 FALSE，Q 首先返 回 TRUE，那么 M 就会被设置为 TRUE。此意味着，没调用一次此功能，Q 便会返回 FALSE，直到 CLK 边界 已上升，随即边界再次下降。

声明如下所示:FTRIGInst : F_TRIG ;

- IL 例子:

```
CAL FTRIGInst(CLK := VarBOOL1)
LD FTRIGInst.Q
ST VarBOOL2
```

- FBD 例子:

- ST 例子:

```
FTRIGInst(CLK:= VarBOOL1); VarBOOL2 := FTRIGInst.Q;
```

7.4 计数器

CTU

功能模块增量器:

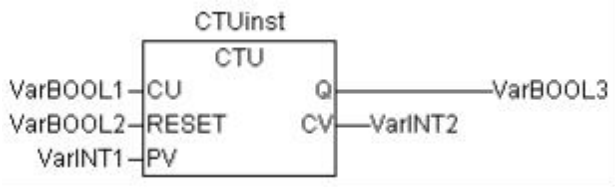
输入变量 CU 和 RESET 以及输出变量 Q 都是 BOOL 类型，输入变量 PV 和输出变量 CV 是 WORD 类型。计数器变量 CV 在 RESET 为 TURE 时被初始化为 0，如果 CU FALSE 到 TURE 为上升边界，CV 将被增加 1，Q 在 CV 大于或等于 PV 上限时，会返回 TRUE。

声明如下所示:CTUInst : CTU ;

- IL 例子:

```
CAL  CTUInst(CU := VarBOOL1, RESET := VarBOOL2, PV := VarINT1)
LD   CTUInst.Q ST   VarBOOL3
LD   CTUInst.CV ST   VarINT2
```

- FBD 例子:



- ST 例子:

```
CTUInst(CU:= VarBOOL1, RESET:=VarBOOL2 , PV:= VarINT1);
VarBOOL3 := CTUInst.Q ;
VarINT2 := CTUInst.CV;
```

CTD

功能模块减数器:

输入变量 CD 和 LOAD 以及输出变量 Q 都是 BOOL 类型，输入变量 PV 和输出变量 CV 为 WORD 类型。

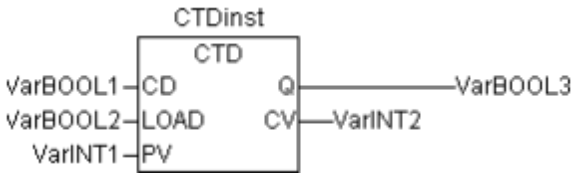
当 LOAD_为 TRUE 时，如果 CV 大于 0(即，它不致让值低于 0).，CV 则会下降 1。当 CVis 等于 0 时 Q 返回 TRUE。

声明如下所示:CTDInst : CTD ;

- IL 例子:

```
CAL  CTDInst(CD := VarBOOL1, LOAD := VarBOOL2, PV := VarINT1)
LD   CTDInst.Q ST   VarBOOL3
LD   CTDInst.CV ST   VarINT2
```

- FBD 例子:



- ST 例子:

```
CTDInst(CD:= VarBOOL1, LOAD:=VarBOOL2 , PV:= VarINT1); VarBOOL3 :=
CTDInst.Q ;
VarINT2 := CTDInst.CV;
```

CTUD

功能模块增量器/减量器

输入变量 CU，CD，RESET，LOAD 以及输出变量 QU 和 QD 都是 BOOL 类型，PV 和 CV 是 WORD 类型。 若 RESET 有效，计数变量 CV 以 0 值进行初始化。若 LOAD 有效，CV 则以 PV 进行初始化。

若 CU 从 FALSE 到 TRUE 边界下降，则 CV 上 1。如果 CD 工 FALSE 到 TRUE 为上升边界，如果这不会使 数值降到 0 以下，CV 则会降 1。

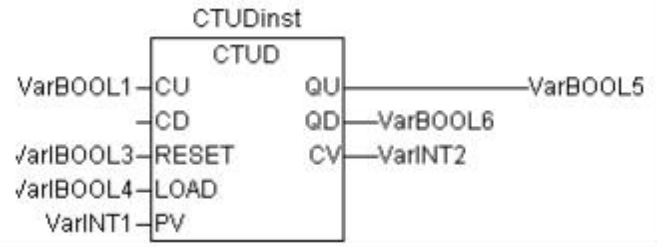
CV 大于等于 PV 时，QU 返回 TRUE。CV 等于 0 时，QD 返回 TRUE。

声明如下所示:CTUDInst : CUTD ;

- IL 例子:

```
CAL  CTUDInst(CU:=VarBOOL2, RESET:=VarBOOL3, LOAD:=VarBOOL4, PV:=VarINT1)
LD   CTUDInst.Q ST   VarBOOL5
LD   CTUDInst.QD ST   VarBOOL5
LD   CTUInst.CV ST   VarINT2
```

- FBD 例子:



- ST 例子:

```
CTUDInst(CU := VarBOOL1, CU:= VarBOOL2, RESET := VarBOOL3, LOAD:=VarBOOL4 ,
PV:= VarINT1);
VarBOOL5 := CTUDInst.QU ; VarBOOL6 := CTUDInst.QD ; VarINT2 := CTUDInst.CV;
```

7.4 定时器

TP

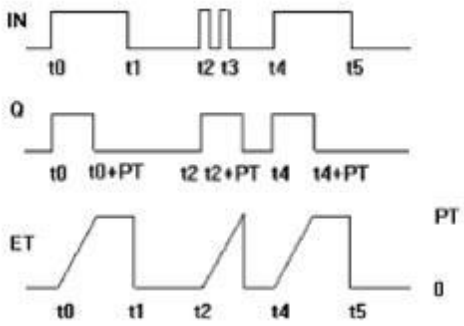
功能块定时器是一个触发器。

TP（IN，PT，Q，ET）意思是:

IN 和 PT 分别是类型为 BOOL 和 TIME 的输入变量。Q 和 ET 分别是类型为 BOOL 和 TIME 的输出变量。 若 IN 是 FALSE，Q 是 FALSE，ET 为 0。

在 IN 变成 TRUE 时，ET 中的时间便开始以毫秒计数，直到其值等于 PT。然后便保持恒定。 当 IN 为 TRUE 并且 ET 小于或等于 PT 时，Q 为 TRUE，。否则 Q 是 FALSE。

Q 返回一个 PT 中已知的时间段信号。 定时器时间顺序的图表显示：

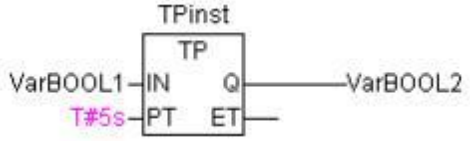


声明如下所示:TPInst ： TP ；

- IL 例子:

```
CAL TPInst(IN := VarBOOL1, PT := T#5s)
LD TPInst.Q
ST VarBOOL2
```

- FBD 例子:



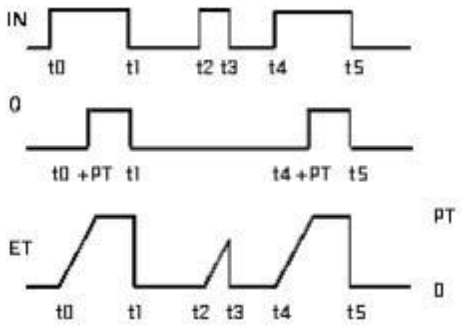
- ST 例子:

```
TPInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 :=TPInst.Q;
```

TON

功能块 Timer On Delay 实现开启延迟。 TON(IN, PT, Q, ET)意思是：IN 和 PT 分别是类型为 BOOL 和 TIME 的输入变量。Q 和 ET 分别是类型为 BOOL 和 TIME 的输出变量。 若 IN 是 FALSE，Q 则是 FALSE，ET 则为 0。

IN 一旦变成 TRUE 时，ET 中的时间便开始以毫秒计数，直到其值等于 PT。然后便保持恒定。当 IN 为 TURE 并且 ET 等于 PT 时，Q 为 TRUE。否则 Q 为 FALSE。这样，当 PT 所示的以毫秒为单位的时间耗尽时，Q 呈现一个上升的边界。 TON 时间行为图表如图：



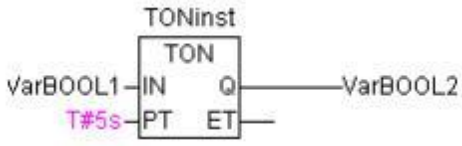
声明如下所示:TONInst : TON ;

- IL 例子:

```
CAL TONInst(IN := VarBOOL1, PT := T#5s)
LD TONInst.Q
ST VarBOOL2
```

288

- FBD 例子:



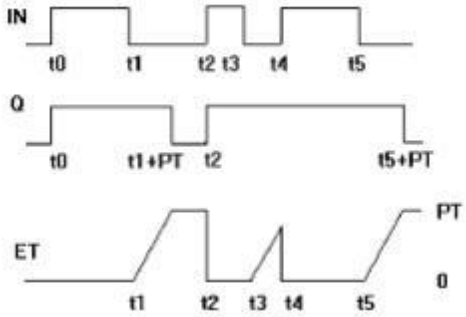
- ST 例子:

```
TONInst(IN := VarBOOL1, PT:= T#5s);
```

TOF

功能块 TOF 实现关闭延迟。

TOF(IN, PT, Q, ET)意思是：IN 和 PT 分别是类型为 BOOL 和 TIME 的输入变量。Q 和 ET 分别是类型为 BOOL 和 TIME 的输出变量。 若 IN 是 TRUE，则输出变量分别为 TRUE 和 0。一旦 IN 变为 FALSE，ET 中时间便开始以毫秒为单位进行计时，直到值等于 PT 为止。然后便保持恒 定。当 IN 为 FALSE 并且 ET 等于 PT 时，Q 为 FALSE。否则 Q 为 TRUE。这样，当 PT 以毫秒为单位所示的时间耗尽时，Q 的边界下降。 TOF 时间行为图表如图：

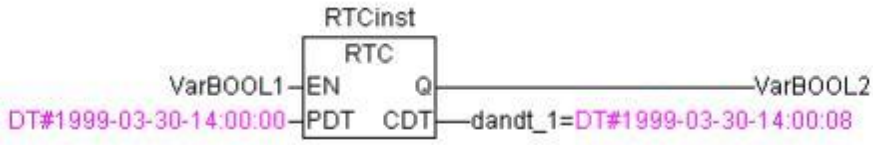


声明如下所示:TOFInst : TOF ;

- IL 例子:

```
CAL TOFInst(IN := VarBOOL1, PT := T#5s)
LD  TOFInst.Q
ST  VarBOOL2
```

- FBD 例子:



- ST 例子:

```
TOFInst(IN := VarBOOL1, PT:= T#5s); VarBOOL2 :=TOFInst.Q;
```

‘

RTC

定时开始，功能块 Runtime Clock 返回当前时间和日期。

RTC(EN, PDT, Q, CDT)意思是：

EN 和 PDT 是类型为 TIME 输入变量，Q 和 CDT 分别是类型为 BOOL 和 DATE_AND_TIME 输出变量。若 EN 为 FALSE 时，输出变量 Q 和 CDT 则分别时 FALSE 和 DT#1970-01-01-00：00：00。一旦 EN 变为 FALSE，PDT 时间已定，并以秒为单位计时。

只要 EN 变成 TRUE，CDT 则复原(见上例图示)。每当 EN 重设成 FALSE，则 CDT 又重设为初始值 DT#1970-01-01-00：00：00。请注意，PDT 的时间只能被上升边界设定。

7.5 Util.lib 库

此库包含可以用于 BCD 转换，位/字节功能，数字辅助功能等各种块的附加集，以作模拟值处理控制 器，信号生成器和功能操控器。

由于其中一些功能和功能块含有 REAL 变量，还有一个名为 UTIL_NO_REAL 的附加库，其中不含有这 些 POU。

7.5.1 BCD 转换

BCD 格式中一个字节含有 0-99 之间的整数。每个十进位的空间占用 4 个位。10 个十进位的空间被存 在 4-7 位之间。这样， BCD 格式则与十六进制表达方式是相似的。唯一不同的是 BCD 格式中只存 0-99 的 数字，而十六进制字节是从 0-FF

示例：整数 51 应转换位 BCD 格式。5 在二进制中为 0101，1 在二进制中为 0001，这样 BCD 字节便为 01010001，此对应的值为$51=81。

BCD_TO_INT

此功能把 BCD 格式的一个字节转换成 INT 值：

此功能的输入值为 BYTE 类型，输出为 INT 类型。

字节应转换的地方，如不是 BCD 格式，其输出结果为 1。

- ST 例子:

```
i:=BCD_TO_INT(73); (* 结果是 49 *)
 k:=BCD_TO_INT(151); (* 结果是 97 *)
l:=BCD_TO_INT(15);(* 输出 -1, 因为它不是 BCD 格式 *)
```

INT_TO_BCD_

此功能把 INTEGER 值转换成 BCD 格式 的一个字节：此功能的输入值为 INT 类型，输出为 BYTE 类型。

数字 255 出现在 INTEGER 应该转换，然而却不能转换为字节的地方。

- ST 例子：

```
i:=INT_TO_BCD(49); (* 结果是 73 *)
k:=BCD_TO_INT(97); (* 结果是 151 *)
l:=BCD_TO_INT(100); (* 错误! 输出: 255 *)
```

7.5.2 位/字节功能

EXTRACT

这功能的输入是一个 DWORD X，以及一个 BYTE N。此输出为一个 BOOL 值，其中含有输入 X 的第 N个位的内容，由此，此功能开始从零位计数。 ST 例子：

```
FLAG:=EXTRACT(X:=81, N:=4); (* 结果: TRUE, 因为 81 的二进制数为 1010001, 所以第四位 是 1 *)
FLAG:=EXTRACT(X:=33, N:=0); (* 结果: TRUE, 因为 33 的二进制数为 100001, 所以第零位是1 *)
```

PACK

此功能可以回送 8 个输入位 B0，B1，…，B7，从类型 BOOL 作为一个 BYTE。功能块 UNPACK 与此功能紧密相关。

PUTBIT

此项功能的输入含有一个 DWORD X，一个 BYTE N 和一个布尔值 B。PUTBIT 把值 B 上从 X 设到第 N 位， 由此从零位计数。

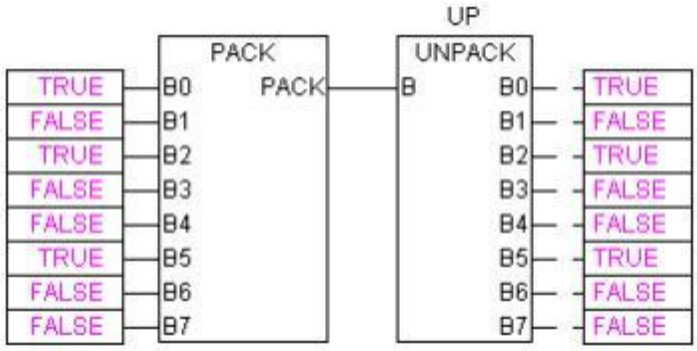
- ST 例子：

```
A:=38;      (* 二进制数为 100110 *)
B:=PUTBIT(A,4,TRUE); (* 结果: 54 = 2#110110 *)
C:=PUTBIT(A,1,FALSE); (* 结果: 36 = 2#100100 *)
```

UNPACK

UNPACK 把输入变量 B 从类型 BYTE 转换为类型 BOOL 的 8 个输出变量 B0，…，B7，这是与 PACK 相对 的。

- FBD 例子：输出：



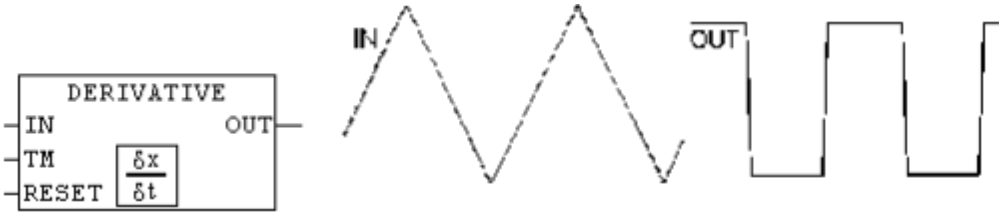
7.5.3 数学辅助功能

微分

微分功能块。

功能值是使用 IN 以 REAL 类型变量表达的。TM 包含在 DWORD 中以毫秒为单位已通过的时间，类型 BOOL 的 RESET 输入可使此功能块通过释放值为 TRUE 而使其重新开始。输出 OUT 的类型为 REAL。

为了取得最佳结果，派生约计使用最后的 4 个值，以便尽量减少输入参数不精确所产生的错误。 FBD 下的功能块：



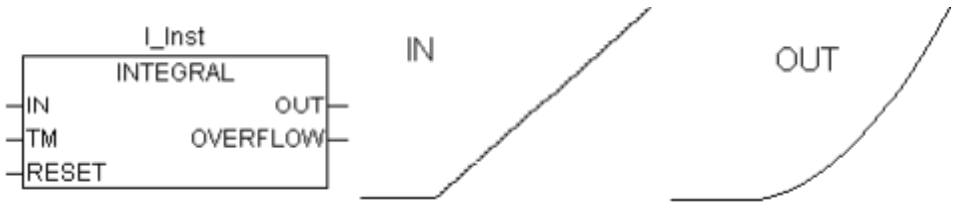
积分

此功能块大致确定功能的积分。

在派生的模拟方式中，功能值的方式是使用 IN 的 REAL 变量。

TM 含有在 DWORD 中以毫秒为单位所记录的时间并且类型为 BOOL 的输入 RESET 可以使功能块以 TRUE 值开 始重新启动。输出 OUT 类型为 REAL。积分为两个步进功能的约数。这些平均值显示为约定的积分数。

FBD 下的功能块： 例如：线性功能的综合：



LIN_TRAFO

此功能块(util.lib)转换一个 REAL 值，此致在已定义有上下值限的值域内为一个相应的 REAL 值，此值 则在另一个已定义的有上下值限的值域内。下面的等式便是此项转换的基础：

(IN - IN_MIN) : (IN_MAX - IN) = (OUT - OUT_MIN) : (OUT_MAX - OUT)

输入变量:

变量	数据类型	描述
IN	REAL	输入值
IN_MIN	REAL	输入值域的下限
IN_MAX	REAL	输入值域的上限
OUT_MIN	REAL	输出值域的下限
OUT_MAX	REAL	输出值域的上限

输出变量:

变量	数据类型	描述
OUT	REAL	输出值
ERROR	BOOL	产生错误：TRUE，如IN_MIN=IN_MAX,或如IN不在输入值域内。

应用例子：

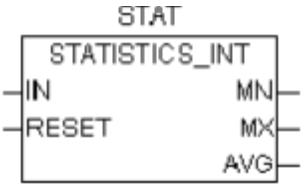
温度传感器提供电压值(输入 IN)。将此温度值转换为摄氏度值(输出值 OUT)。输入（伏特）值域已由 IN_MIN =0 和 IN_MAX＝10 定义的上下限。其输出(摄氏度)值域则由 OUT_MIN=-20 和 OUT_MAX=40 为上下限。这 样，输入为 5 伏的温度其摄氏度的结果为 10 度。

STATISTICS_INT

此功能块计算部分标准的统计学值。

输入值 IN 为 INT 类型。当布尔输入变量 RESET 为 TRUE，所有的值都会被重新初始化。

输出 MN 包含 IN 中的最小值和最大值的 MX。AVG 描述的是平均值，那也是所期望的 IN 值，三个输出变量都是 INT 类型。 FBD 下的功能块:



STATISTICS_REAL

此功能块STATISTICS相呼应，除了其输入变量IN，像MN，MX，AVG都是REAL类型。

VARIANCE

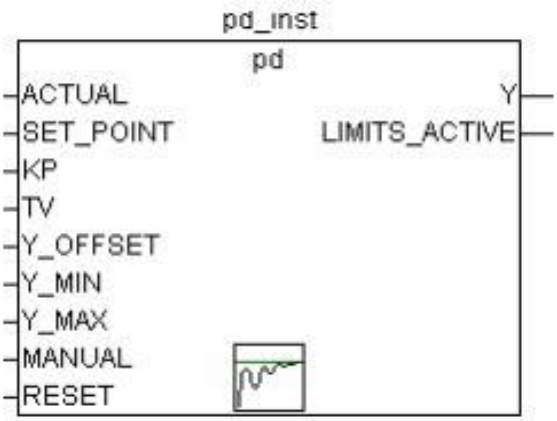
VARIANCE 计算已键入值的变异。

输入变量 IN 为 REAL 类型，RESET 为 BOOL 类型，输出 OUT 也是 REAL 类型。此块计算的输入值的变 异。VARIANCE 可以用 RESET＝TRUE 重新设置。标准偏差可以用 VARIANCE 的平方根的方法很容易地被计算出来。

7.5.4 控制器

PD

PD 控制器功能块:



ACTUAL（当前值）和 SET_POINT(期望或额定值)以及 KP，均衡系数，都是 REAL 类型的输入值。TV 为 DWORD 类型，含有以秒为单位的(如 0.5 秒代表 500 毫秒)派

生活动时间。Y_OFFSET，Y_MIN 和 Y_MAX 为 REAL 类型，被用于已指定域内操纵量的转换。类型为 BOOL 类型，被用于重置控制器。

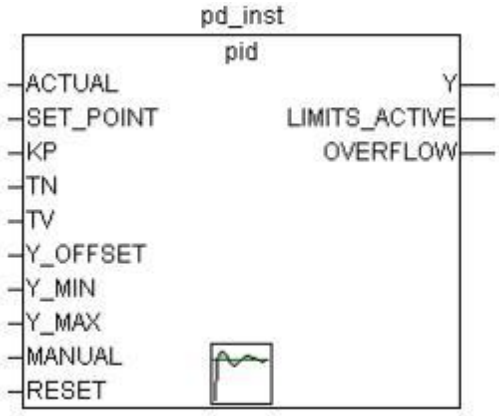
Y 也被限定在所允许的 Y_MIN 和 Y_MAX 的范围之间。如果 Y 超出此范围，作为布尔输出变量的 LIMIT_ACTIVE 便成为 TRUE。如果想让操纵量无限制，Y_MIN 和 Y_MAX 设为零。

若 MANUAL 为 TRUE，那么调节器被暂停使用，即 Y 不被更改（被控制器），如果 MANUAL 变为 FALSE， 此时则会重新初始化控制器。

P 控制器可通过设置 TV，很容易使其生成一个固定的值零。

PID

PID 控制器功能块:



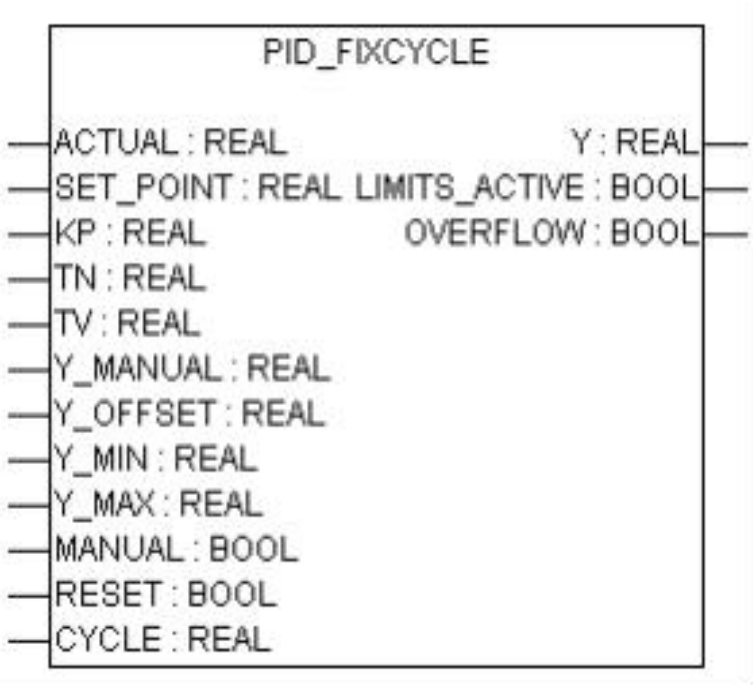
此功能不像 PD 控制器功能，它含有一个更深层次的 DWORD 输入变量 TN，以秒为单位(如 0.5 秒代表 500 毫秒)调整时间。输出操纵量（Y）还是 REAL 类型，并不像 PD 控制器那样含有一个附件的积分部分：

Y = KP x (D + 1/TN ?edt + TV dD/dt) + Y_OFFSET

PID 控制器可以很方便地通过将 TV 设置成零转为 PI 控制器。因为有了附加的积分部分，所以控制器参 数输入出现错误就会造成溢出，条件是错误的积分变得过大。因此，为了安全起见，便会把布尔输出的 变量 OVERFLOW 调用出来。此时其值应该是 TRUE。同时，控制器暂停，只有重新初始化时才会被再次激活。

PID_FIXCYCLE

PID_FIXCYCLE 控制器功能块:



此功能块与 PID 控制器相应，除了周期不是有内部功能自动测量，而是被输入变量 CYCLE（以秒为单位） 进行设定。

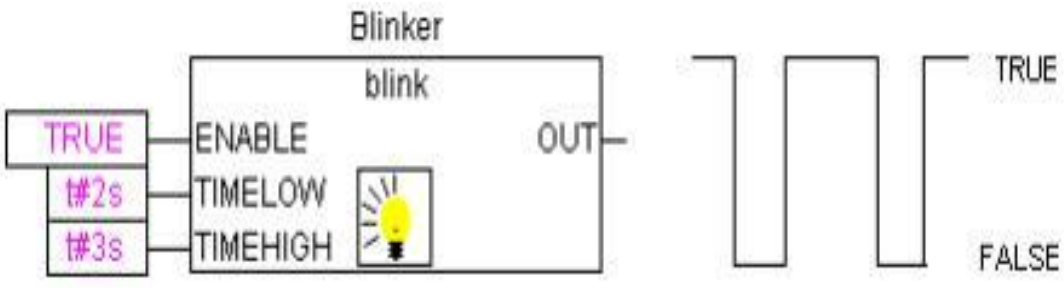
7.5.5 信号生成

BLINK

BLINK 功能块可生成脉冲信号。其输入变量由布尔类型的 ENABLE 以及 TIMELOW 和 TIMEHIGH 时间类 型组成的。输出变量 OUT 类型为 BOOL。

如果 ENABLE 被设为 TRUE，BLINK 便开始生效，将时间输出变量 TIMEHIGH 设为 TRUE，然后将时间 TIMELOW 的变量值设为 FALSE。

- CFC 例子:



FREQ_MEASURE

此功能块用于测量（平均）布尔输入信号的频率。可以详细列出应当平均为多少个时间段。一个时间段 便为两个输入信号的两个边界的上升之间的时间。

输入变量:

变量	数据类型	描述
IN	BOOL	输入信号
PERIODS	INT	时段数量，即，边界上升的间隔时间，应当计算输入信号的平均频率。可能的值为1-10。
RESET	BOOL	把所有参数重置为0

输出变量:

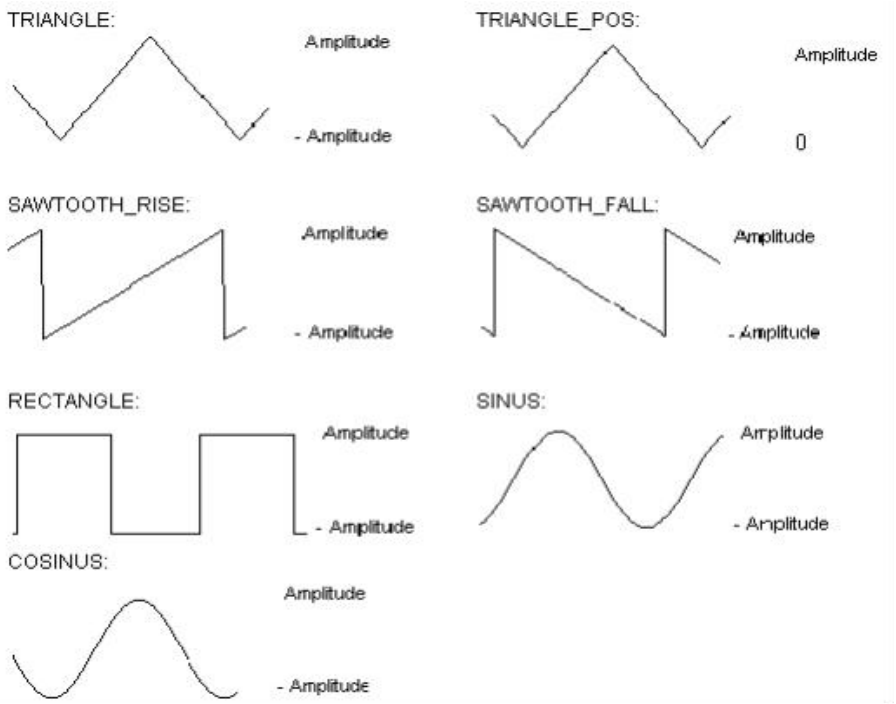
变量	数据类型	描述
OUT	REAL	得到频率[赫兹]
VALID	BOOL	首次测量之前为FALSE，或者如果时间长度>3*OUT（显示输入变量出了问题）

GEN

功能生成器生成典型的周期功能:

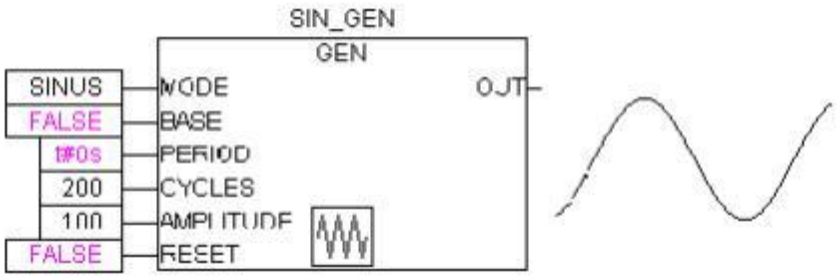
输入变量为一个由 MODE 组成，其中包括预定的计数类型 GEN_MODE，BOOL 类型的 BASE， TIME 类型 的 PERIOD，组成部分中已有两个 INT 型的值 CYCLE 和 AMPLITUDE 以及布尔 RESET 输入变量。

MODE 说明所生成功能利用列举值 TRIANGLE 和 TRIANGLE_POS 传送两个三角形功能。SAWTOOTH_RISE 为一个上升功能，而 SAWTOOTH_FALL 为一个下降的锯齿，RECTANGLE 为一个矩形信号，SINE 和 COSINE 分别为正弦和余弦:



BASE 定义周期是否真的与所定义的时间（BASE＝TRUE）有关，或者它是如何与 PERIOD 或 CYCLE 定义相 应的周期时间。AMPLITUDE 从细节方式定义所生成的功能的周期变量的最大绝对值。此功能生成器在 RESET＝TRUE 时就会再次被设为 0。

- FBD 例子:



7.5.6 功能操作器

CHARCURVE

此项功能块用于在线性功能中一一表现值:

INT 类型中的 IN 要用值作操作源进行操作。BYTE N 指定点数，此点定义表现功能。此特征线随后便 用类型 POINT 中的 P 在一个 ARRAY P[0……10]中生成，其中的 POINT 为一个基于 2 个 INT 值（X 和 Y） 两个值的一个结构。输出的组成部分有 INT 中的 OUT。操作的值以及 BYTE ERR 此项主要在必须时显示错误。ARRAY 中的点 P[0]…P[N-1]必须根据其 X 值进行分类，否则，ERR 便会接收到值 1。如果输入的 IN 不在 P[0].X 和 P[N-1].X，则 ERR＝2 并且含有相应限值 P[0].Y 或 P[N-1].Y。如果 N 不在所允许的 2- 11 之间的值域内，那么 ERR＝4。

- ST 例子:

首先，ARRAYP 必须在页首被定义:

```
VAR
...
CHARACTERISTIC_LINE:CHARCURVE;
KL:ARRAY[0..10] OF POINT:=(X:=0,Y:=0),
 (X:=250,Y:=50),
(X:=500,Y:=150),(X:=750,Y:=400),7((X:=1000,Y:=1000));
COUNTER:INT;
...
END_VAR
```

再提供一个值一直增加的情况下 CHARCURVE 的例子: COUNTER:=COUNTER+10; CHARACTERISTIC_LINE(IN:=COUNTER,N:=5,P:=KL); 结果跟踪显示说明其结果:

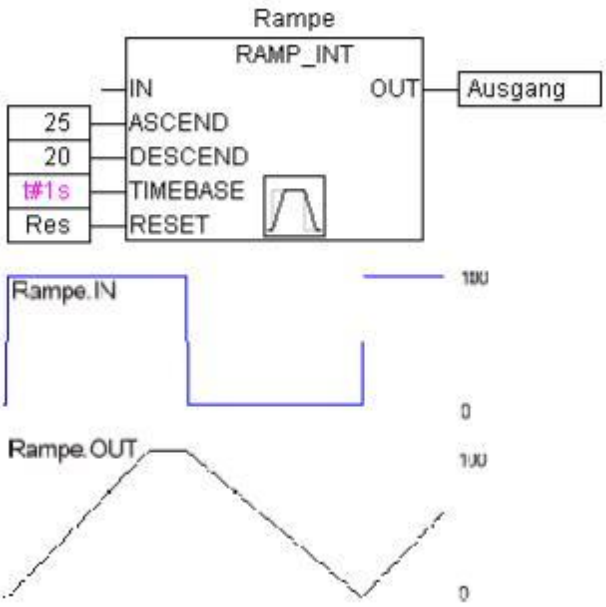
RAMP_INT

输入变量由三个 INT 值的 OUT 变量组成：IN，功能输入以及 ASCEND 和 DESCEND，每个间隔之间的最 大增或减量，它又是被 TIME 类型中的 TIMEBASE 来定义的。将 RESET 设为 TRUE 会导致 RAMP_INT 被重新 初始化。

INT 类型中的 OUT 输出变量中含有上升和下降的有限功能值。

把 TIMEBASE 设为 t#0s，ASCEND 和 DESCEND 虽与时间间隔无关，但是却能保持不变。

CFC例子:



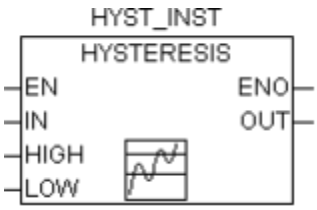
RAMP_REAL

RAMP_REAL 与 RAMP_INT 中的功能一样，不同的是，其中的输入变量为 IN，ASCEND， DESCEND 和输出变量 OUT 皆为 REAL 类型。

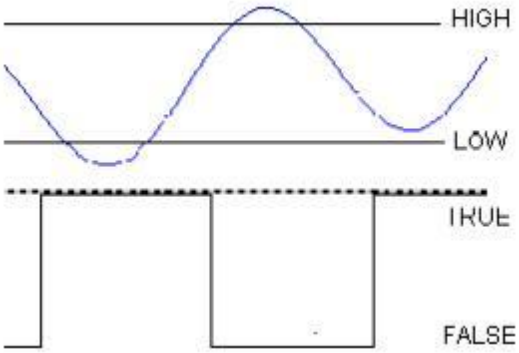
7.5.7 模拟值的处理

HYSTERESIS

此功能块的输入变量包括三个 INT 值：IN，HIGH 和 LOW。输出变量 OUT 为 BOOL 类型。



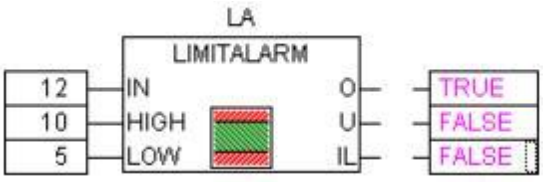
如果 IN 值低于 LOW 的限值，OUT 则变成 TRUE。若 IN 超过了上限 HIGH 的值，便会显示 FALSE 的结果。 说明示例：



限制警告

此项功能块说明输入变量的值是否在设定范围内，如果不在范围内便超出了限制范围。输入值 IN， HIGH 和 LOW 都是 INT 类型，而输出变量 O，U 和 IL 则都是 BOOL 类型。

如果上限 HIGH 被 IN 超过，O 便变为 TRUE，而如果 IN 低于 LOW，U 就会编程 TRUE。如果 IN 介于 LOW和 HIGH 之间 IL 则为 TRUE。 FBD 例子：结果：



7.6 AnalyzationNew.lib 库

此程序库提供表达式的分析模块。如果组成表达式为 FALSE，其组成部分可以进行评估，并加注到 这个结果中。在 SFC-Editor 中 SFCErrorAnalyzationTable 标记，自动使用此功能对运算过程中的表达 式进行分析。

分析示例:

b OR NOT(y < x) OR NOT (NOT d AND e) 功能:

下面的变量用于所有模块:

InputExpr: BOOL, 要进行分析的表达式。 DoAnalyze: BOOL, TRUE 开始分析。

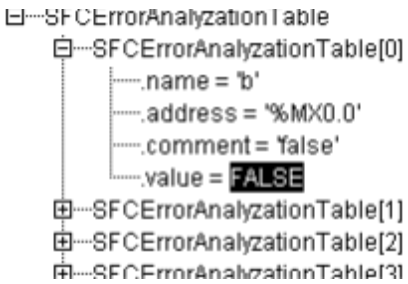
ExpResult: BOOL, 表达式当前值。 分析结果输出不同:

AnalyzeExpression 复原字符串中表达式的成分，则将其加注在整个 FALSE 中。AppendErrorString 功能用于此目的，用“|”分开输入字符串的特定成分。

OutString：STRING，分析结果，表达式相关成分的顺序（如：y<x|d）

AnalyseExpressionTable 写出表达 式的成分，并加注到总值 FALSE 之中。每个成分都由 structureExpressionResult 加注如下信息：姓名，地址，备注，（现）值。

例如：



AnalyseExpressionCombined 将 AnalyzeExpression 和 AnalyseExpressionTable 的功能性合并。

8 CoDeSys 运动控制模块

8.1 运动控制的实现方式

1.运动控制和逻辑控制单独控制

（1）昂贵 ， 需要两个设备

（2）同步/通讯需要大量资源

（3）使用两种不同编程工具

2.一个设备 ， 但是使用两个编程工具：

（1）使用两种不同编程工具:

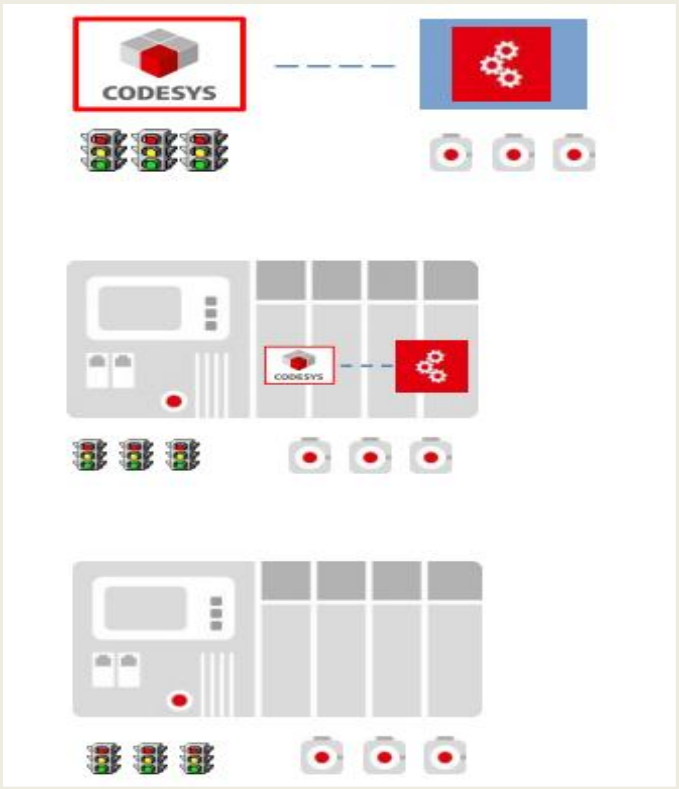
（2）同步/通讯需要大量资源

3. 一个设备 ， 一个编程工具

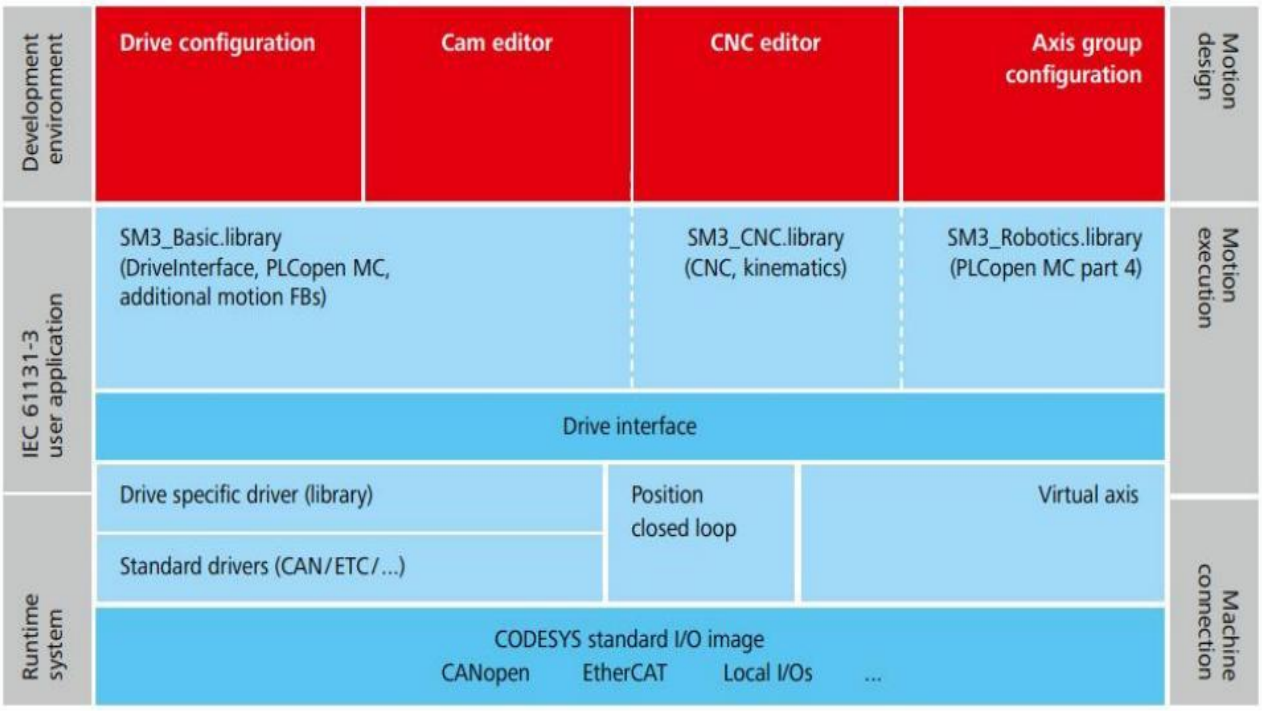
（1）经济

（2）使用一个编程工具编程

（3）运动和逻辑控制在同一进程中 ， 所以同步/通讯不需要额外资源



8.2 架构总览



8.3 控制方式

- 用于伺服控制

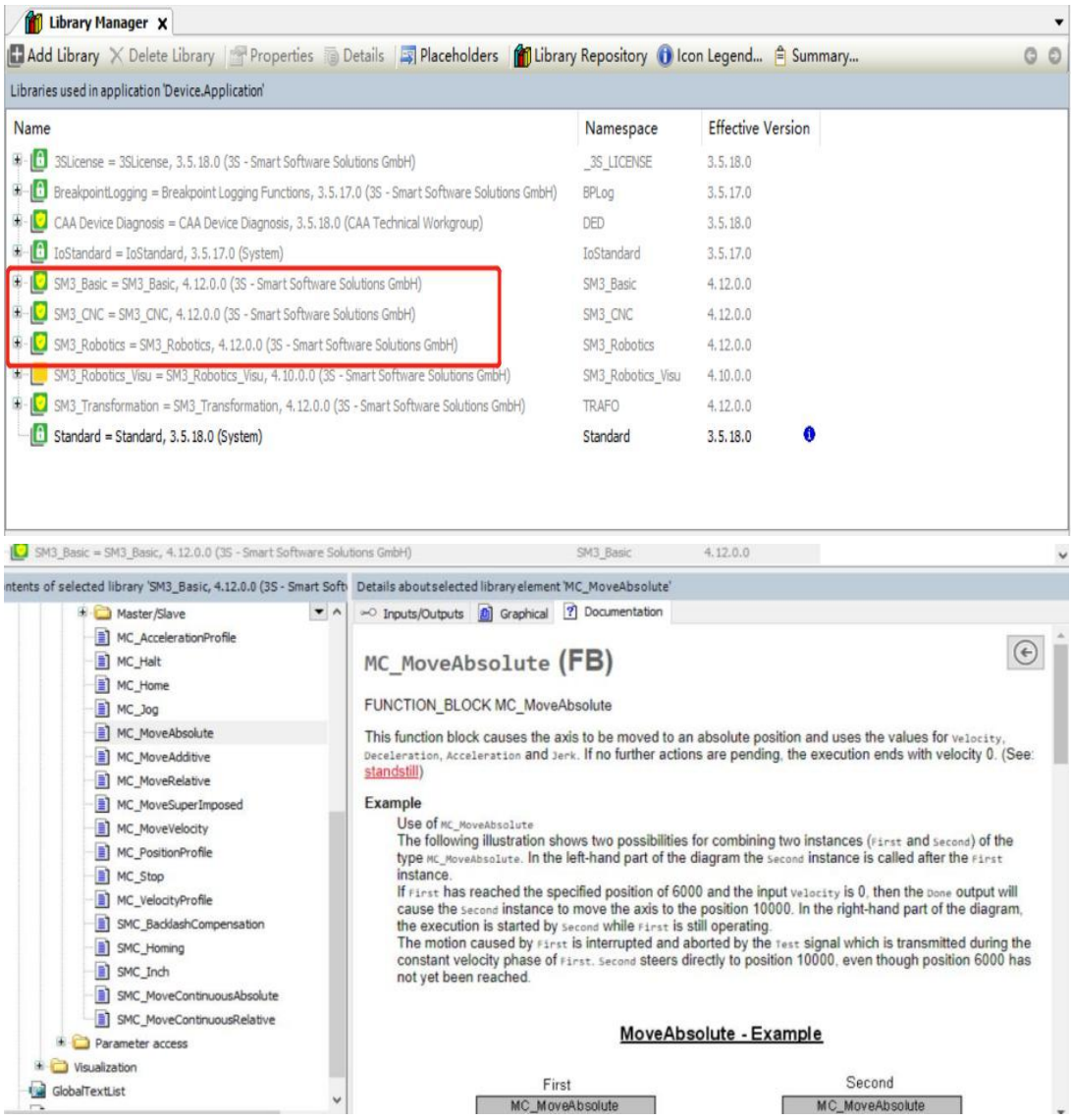
- 用于步进控制

- 用于变频器控制

8.4 控制功能

Feature/Product	CODESYS SoftMotion Light	CODESYS SoftMotion	CODESYS SoftMotion CNC+Robotics
单轴运动	✓	✓	✓
多轴同步运动	✗	✓	✓
命令伺服驱动或者步进电机	✓	✓	✓
控制器中位置控制	✗	✓	✓
电子凸轮	✗	✓	✓
电子齿轮	✗	✓	✓
CNC 应用	✗	✗	✓
Robotic 应用	✗	✗	✓
运动规划编辑器 / 配置器	✗	CAM 编辑器	CNC与轴组编辑器
协同运动的运动学模型	✗	✗	多种运动学模型
需要的资源	低	高	高

8.5 运动控制库



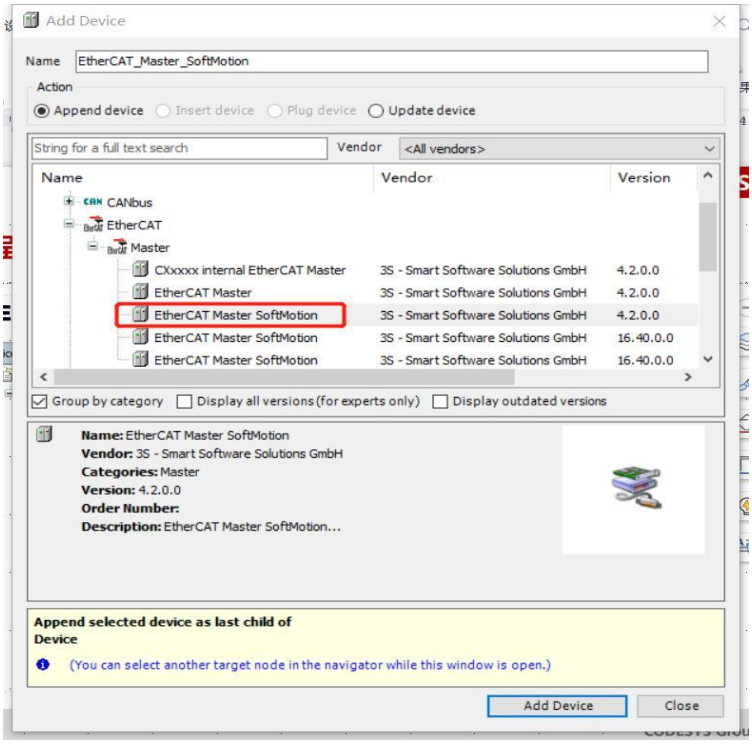
8.6 让轴动起来

8.6.1 选择支持 SoftMotion 的设备

如SoftMotion RTE

8.6.2 添加 EtherCAT 主站和从站

- 1.选择：->EtherCAT Master SoftMotion

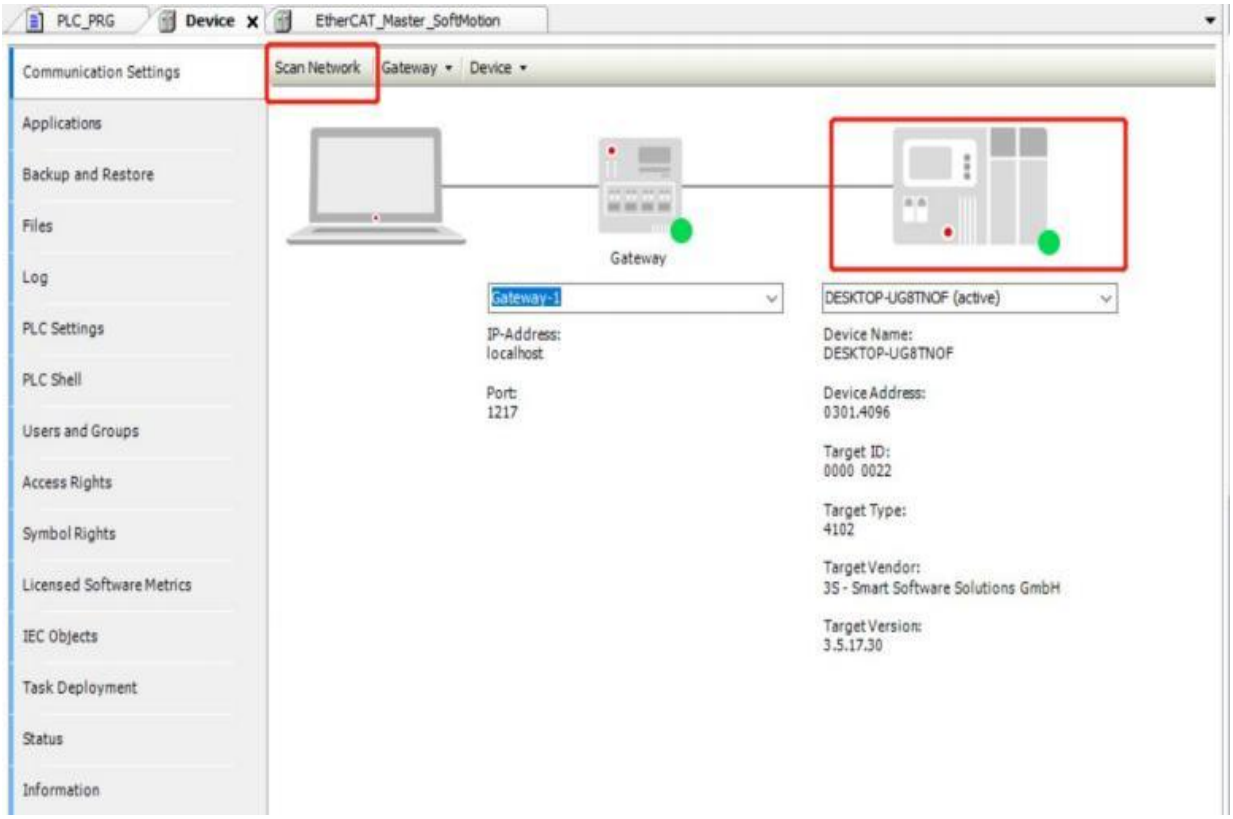


注：需选择带SoftMotion的EtherCAT主站

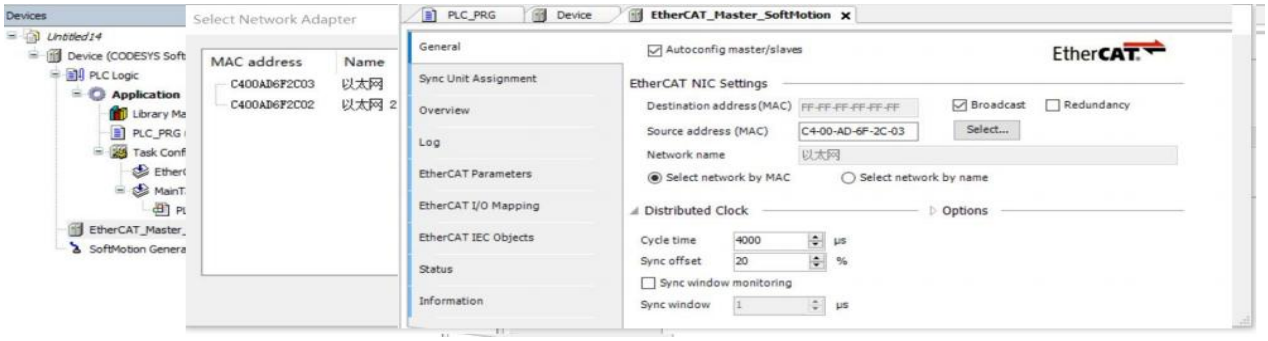
- 2.添加EtherCAT主站后 ，将自动增加EtherCAT_Task

注意：切勿删除该自动添加的EtherCAT_Task， EtherCAT主站的运行依赖于EtherCAT_Task

- 3.扫描设备 ，建立同目标控制器的连接



- 4.选择EtherCAT主站对应的网卡。



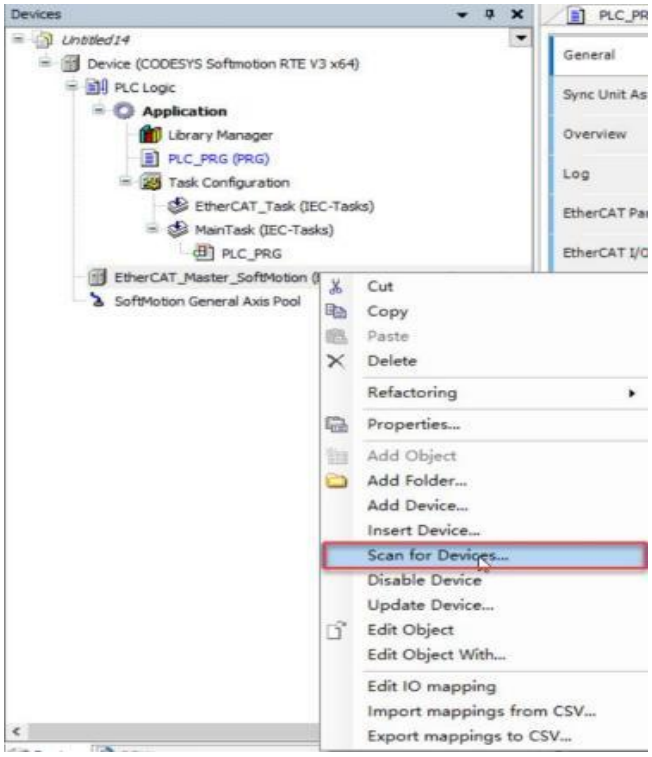
注： RTE需选择适配了CODESYS网卡驱动的网卡。

- 5.下载程序

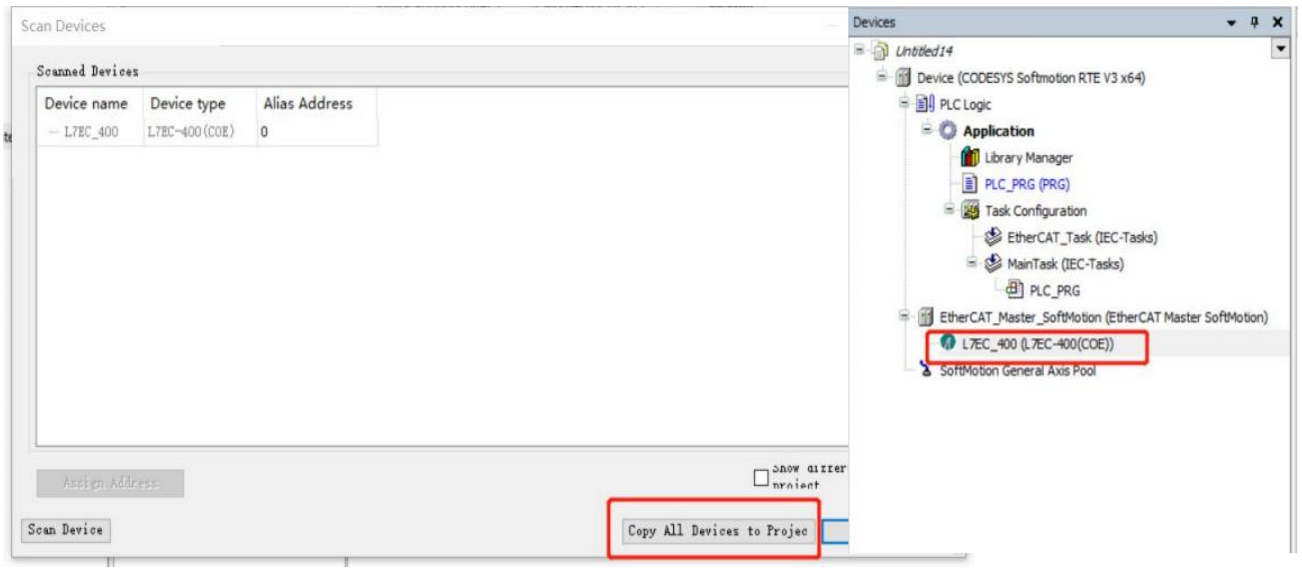
注：只有先下载配置了EtherCAT主站的程序 ，才能扫描连接的EtherCAT从站

- 6.退出登录

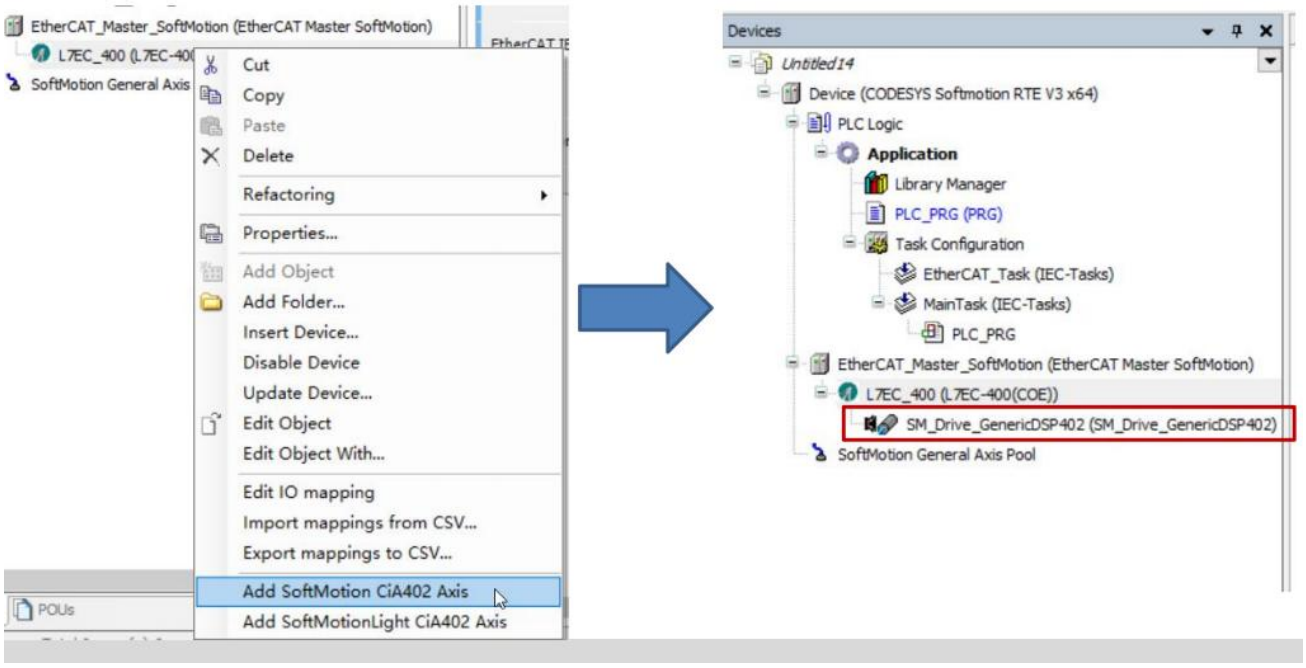
- 8. 扫描从站



- 8. 将扫描到的从站添加到设备树

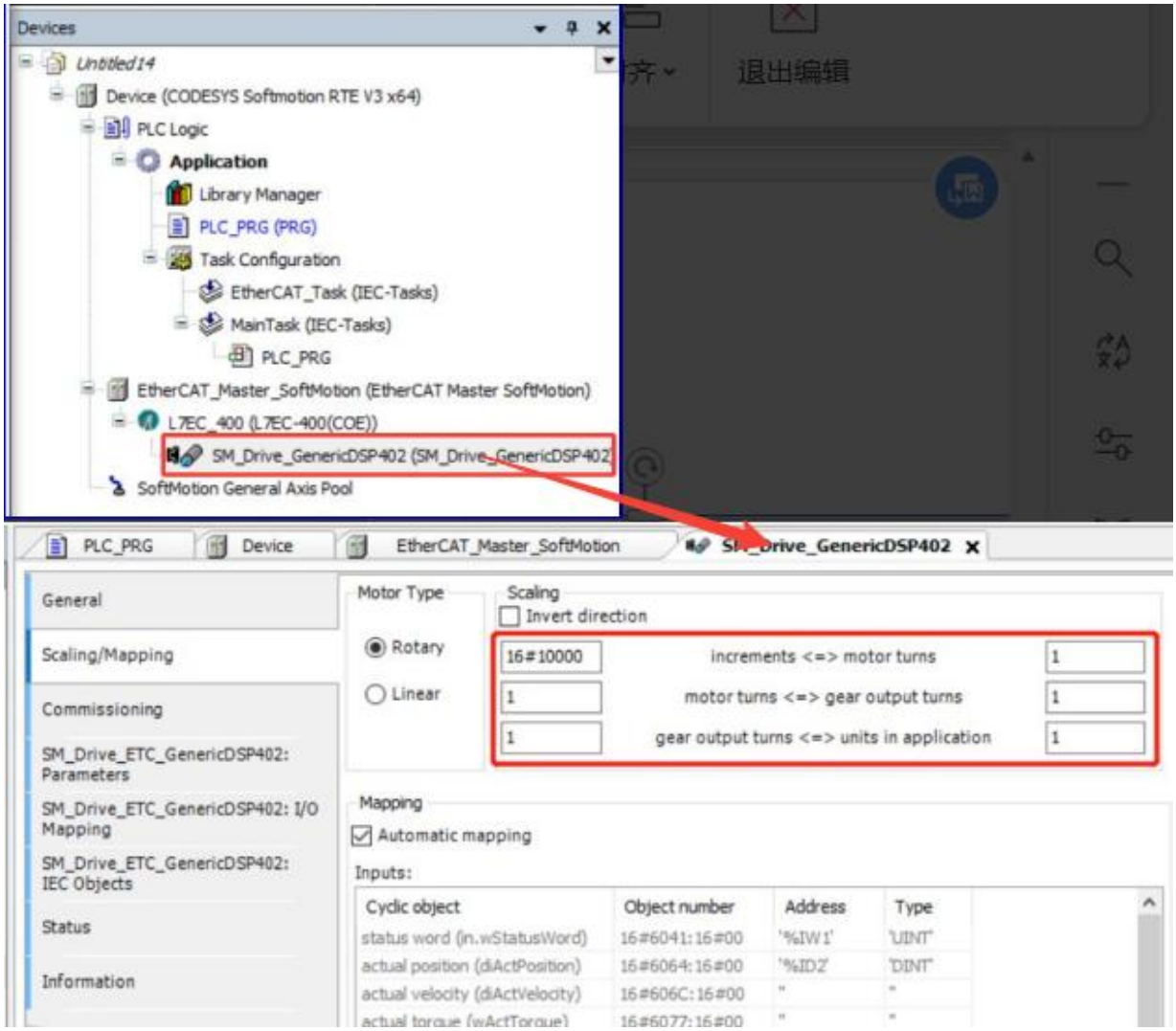


8.6.3 添加标准的 402 轴

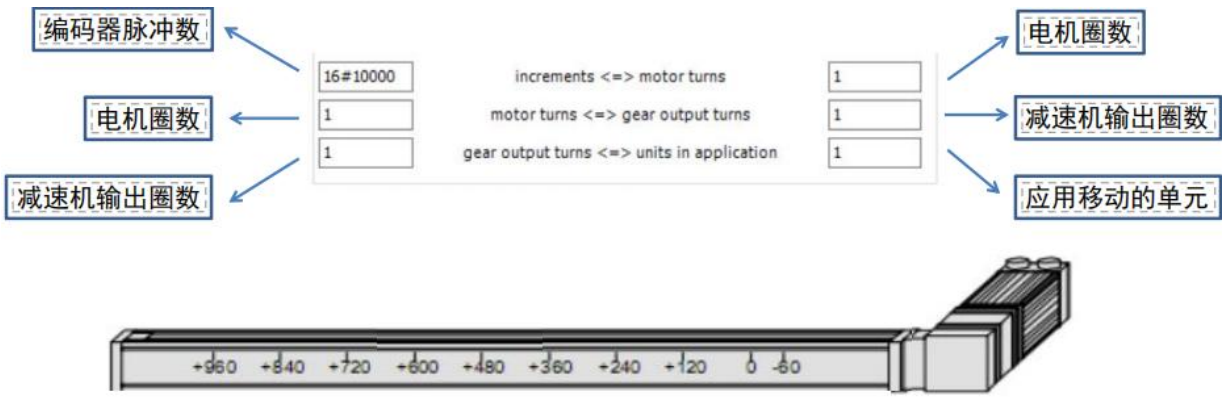


8.6.4 配置电子齿轮

- 1.配置电子齿轮比



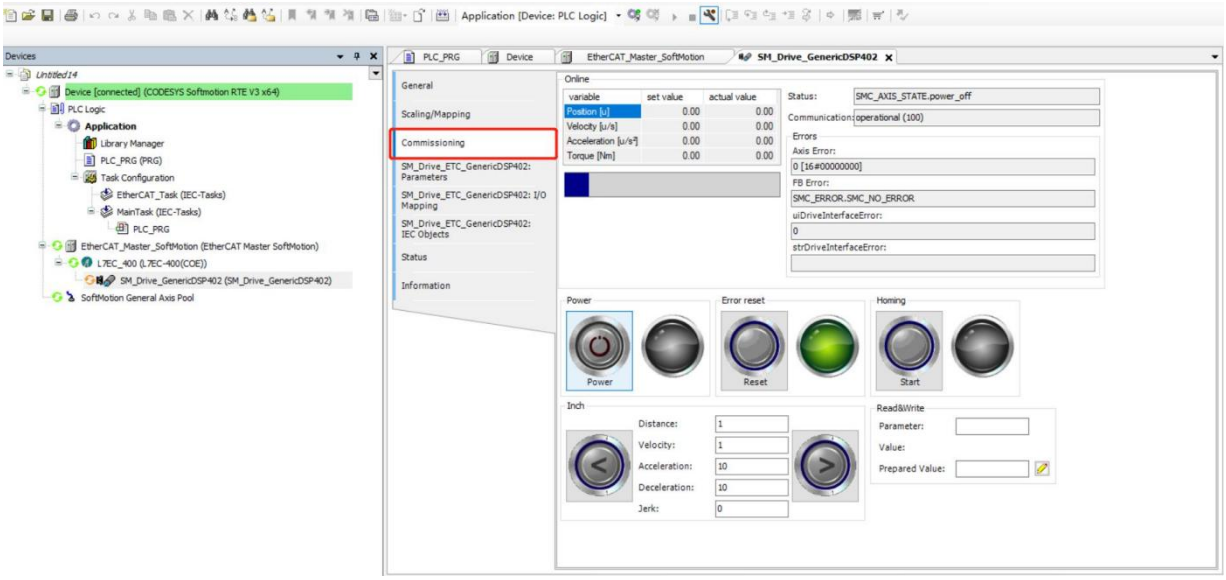
- 2.配置电子齿轮比



8.6.5 在线配置模式

- 1.点击菜单栏的配置按钮

- 2.进入配置参数页面

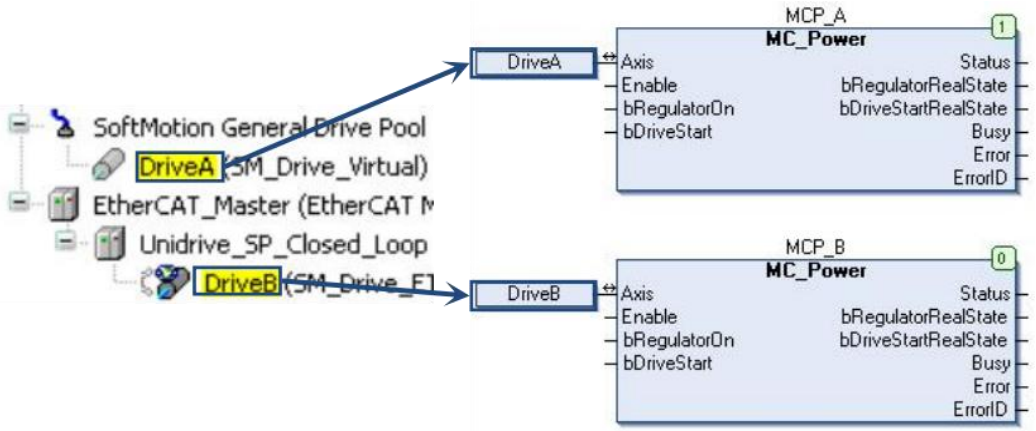


8.7 轴的基础应用

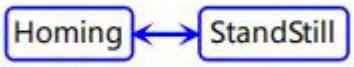
8.7.1 轴使能

MC_POWER：

- 1.启动电源级(bRegulatorOn引脚)---将伺服电机给使能上电；
- 2. 禁用快速停止机制(bDriveStart引脚)



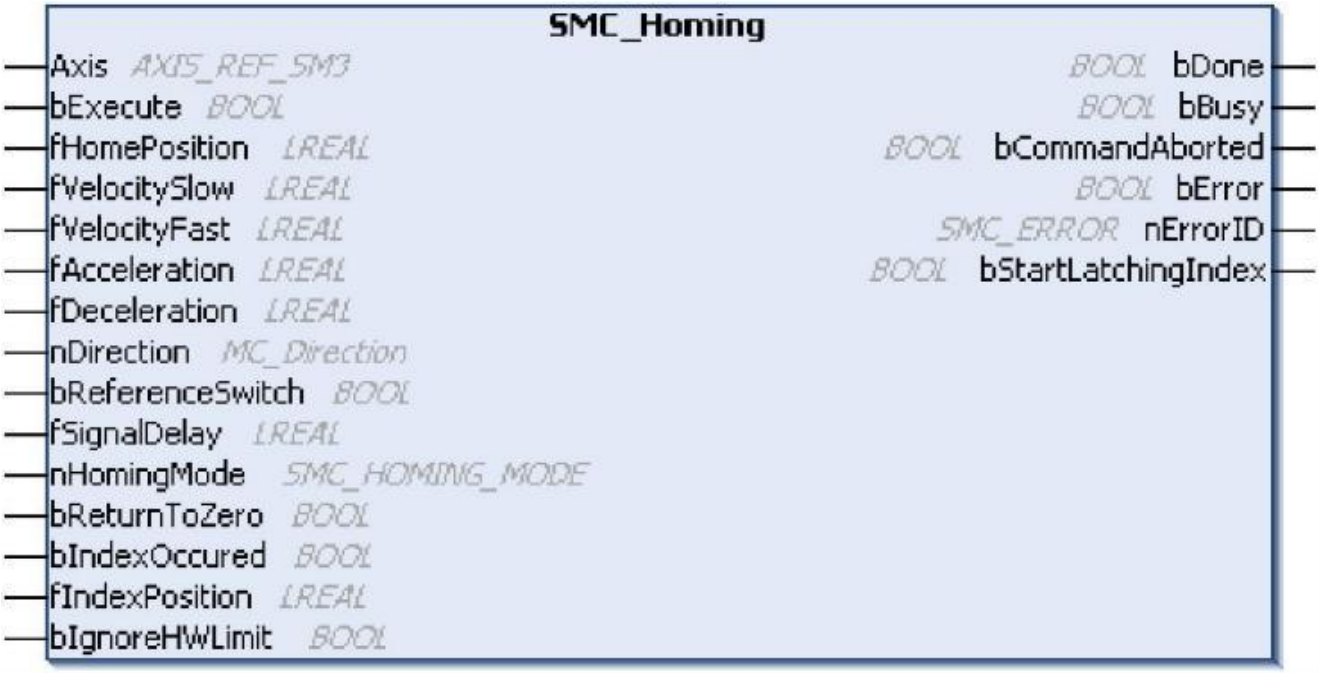
8.7.2 轴回原点

MC_Home： 

- 1.驱动器控制回零(ordered homing)

注：.如果驱动内不支持回零模式，就要使用SMC_Homing

- 2.PLC回零

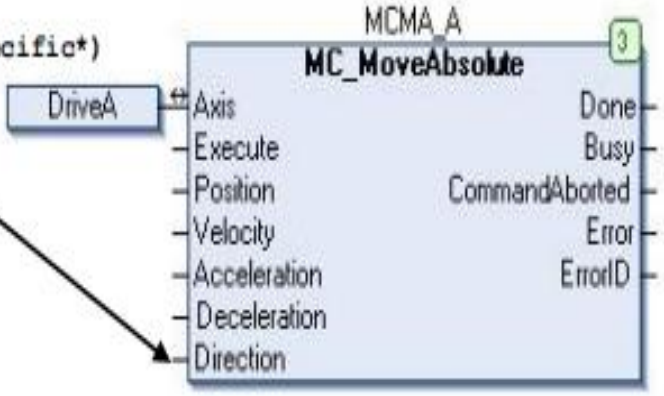


8.7.3 绝对移动

MC_MoveAbsolute：

- 绝对定位
- 方向设定仅用于旋转轴：方向模式：正、负、 电流、最短、 最快

```
TYPE MC_DIRECTION :(
        fastest:=3,(*vendor specific*)
        current:=2,
        positive:=1,
        shortest:=0,
        negative:=-1);
END_TYPE
```



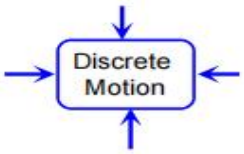
8.7.4 相对移动

MC_MoveRelative：

- 相对于上一个位置进行相对运动.

8.7.5 运动叠加

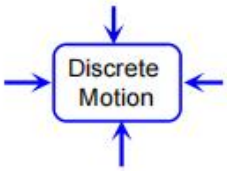
MC_MoveSuperImposed：



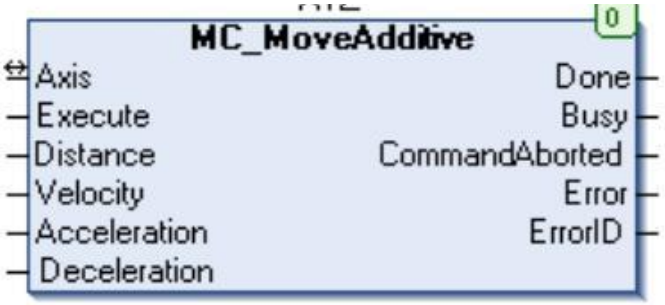
- 在原来运动基础上叠加速度和位置，不 会打断原功能块运行状态

8.7.6 运动添加

MC_MoveAdditive：

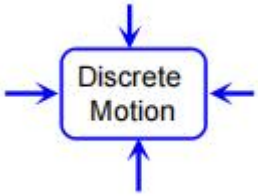


- 只叠加位置，速度以MC_MoveAdditive的设定值运行



8.7.7 其他运动功能块

MC_PositionProfile:



- 设定时间和位置的参数，在规 定的时间内走规定的位置

9 关于编译错误和警告

如果在工程 编译过程中出现错误，将在 信息窗口出现提示。在此窗口按 <F4> 跳到消息的下一行，与此相关的POU将会打开。在错误和警告之前有一个唯一的编号。在消息窗口选择了 一个消息行，按 <F1>打开相应的在线帮助窗口。

1100

未知库中的功能名‘<name>’使用一个外部库。请检查在 hex 文件中定义的所用功能是否也定义在 lib 文件内。

1101

" 不能识别的符号‘<symbol>’"代码生成器期望有一个名称为<symbol>的 POU。它没有在目标项目中定义。并使用这个名称定义了一个功能/程序。

1102

"对符号‘<symbol>’的无效接口"代码发生器期望有一个名为<symbol>的功能和恰好是一个标量的输入，或一个名为<symbol>和无输入或输出的程序。

1103

"在代码地址‘<address>’的常数‘<name>’改写一个 16K 页的边界!"超过 16K 页边界的一串常数。系统不能处理这种边界。这取决于运行时的系统，通过目标文件中的一个登录项是否能解决这个问题。请与 PLC 制造商联系。

1200

"在任务调用的名为'<name>'的POU 中的存取变量内容在参数列表中未获更新" 在任务配置中调用的任务块中所使用的变量将不会列在交叉参考列表中。

1300

"文件找不到'<name>'名"指示全局变量的文件不存在。请检查其路径是否正确。1301"分析库未找到。分析代码无法生成"系统使用了分析功能，但相应的库.lib 没找到。将它加入到库管理程序内。

1302

"新添加了外部引用功能，因此不能使用在线变更功能"由于最后装载，你链接的一个库它所包含的功能尚未在运行时系统中引用。为此，你应装载整个项目。

1400

"未知的编译指令'<name>'，编译器忽略"这个附注不受编译程序支持。受支持的命令见关键字的附注。

1401

"名为'<name>'的结构未定义任何元素"此结构未定义任何元素，但是系统仍然会分配 1B 的内存容量。

1410

"功能中定义本地变量时'RETAIN'和'PERSISTENT'关键字无效"系统将按照一般内部变量处理。

1411

"变量配置中定义的变量'<name>'未在任何任务调用中得到更新"没有在任何调用中引用变量的高水平的例子。因此其不能在过程镜象中被拷贝。

变量配置:

```
VAR_CONFIG
plc_prg.aprg.ainst.in AT %IB0 : INT;
END_VAR
plc_prg:
index := INDEXOF(aprg);
```

程序 aprg 已经引用但是没有调用. 因此 plc_prg.aprg.ainst.in 永远得不到 %IB0 的实际值

1412

"编译指令{pragma name}出现未知代号"用户使用的编译指令包含有在该场合不能正确使用的指令。详情请参看 CoDeSys 在线帮助。

1413

"'<Name>'在名字列表中的索引非法，此索引将被忽略"在编译指令中使用了不存在的参数列表。请参看参数管理器获取当前列表。

1414

名为'<name>'的编译指令定义太多组成部分编译指令包含的定义（指括号里面的内容）超过了相应数组、功能块或者结构所能容纳的元素个数。

1500

"表达式未包含赋值。此语句无代码产生"程序未使用该表达式的运算结果，因此该语句无代码产生。

1501

"字符串常量的内容被改写"在 POU 中定义的字符串常量的内容不允许更改，因为系统忽略其大小检查。

1502

"POU 与其定义的某个变量重名。导致 POU 不能被正确调用" POU 与其定义的某个变量重名。

例如:

PROGRAM a

...

VAR_GLOBAL

a: INT; END_VAR

...

a; (* 没有 POU a 被调用，但是变量已经加载 *)

1503

"名为'<name>'的 POU 无输出，系统设定其输出为'TRUE'"连接 FBD 或 KOP 的 POU 无输出。 系统自动将其设定为 TRUE。

1504

"?lt;name>?(?lt;number>?: 逻辑表达式的此分支将不会被执行，因为整个表达式的值已可确定"逻辑表达式的此分支将不会被执行，因为整个表达式的值已可确定 例如:IF a AND funct(TRUE) THEN ....如果 a 是 FALSE 则 funct 不被调用。

1505

"'<name>'分支有歧义，可能不会被执行"POU 的第一个输入为 FALSE，由于这个原因旁边的第二个输入将不被执行。

1506

"变量与某个本地动作同名，将导致该动作不能被正确调用"重命名变量或行为。

1507

"创建的实例与某个功能名同名，将导致该实例不能被正确调用"在 ST 编程语言中，如果调用与某功能同名的实例，系统将执行同名功能而不是实例。

1550

"名为'<name>'的 POU 在一个执行网络中被多重调用，可能导致不可预料的结果"检查，对 POU 的多重调用是否是必要的。通过多重的调用可能导致不可预料的后果。

1600

"不明确打开 DB （可能产生错误的代码)."没有表明 POU 打开的最初西门子程序。

1700

"输入框未联接上"在 CFC 中有输入框未连接上，将不能产生执行代码。

1750

"步 '<名称>': 设定的时间下限超过时间上限"请使用该步的'步属性'对话框进行时间的正确设定。

1751

"'<Name>'变量的使用警告：该变量使用于固定代码中并且影响步的执行顺序" 推荐对该变量重命名以使其具有唯一的标识符，从而避免负面影响。

1800

"非法的监控表达式'<Name>'"可视化元素中包含了无法监控的表达式，检查变量名称和占位符的替换是否无误。

1801

"表达式不能作为输入变量使用"可视化组件的输入框使用了组合表达式，请使用单个变量替换之

1802

"名为'<Name>'的位图文件未找到"

1803

"网页可视化和目标平台可视化不支持打印操作"

1804

"目标平台不支持'<name>'型字体"

1805

"要使用可视化组件实现PLC 趋势数据的存储应先进行相关的设定"

1806

"应对'目标设定'中的'PLC 报警处理'选项进行设定"

1807

"<name> (<number>): 针对可视目标没有警告消息窗口"

1850

"映射在%IB<number>位置的输入变量在任务'<name>'中使用，但在另外任务中更新"

1851

"映射在%IQ<number>位置的输出入变量在任务'<name>'中使用，但在另外任务中更新"

1852

"CanOpenMaster 功能块在任务'<name>'事件中未能实现循环调用！若要实现其循环调用，请使用'模块参数'对话框正确设置运行参数并更新任务!"

1853

"PDO(index: '<number>')可能在任务'<name>'事件中未能实现周期性更新"

1900

"POU'<name>'（主程序）在库中找不到；当工程作为一个程序库使用时，主入口程序不可用" 当项目作为库使用时，不能利用“启动 POU”

1901

"入口变量和变量配置表未存入相应库文件中!"

存取变量和变量配置没有保存在库内。

1902

"'<name>'库不支持当前硬件平台类型!"

库的对象文件是为其他设备而生成的。

1903

"'<name>'库是非法库；该库文件不符合实际目标平台要求的格式"

文件没有实际目标所需要的格式。

1904

" '<name>'常量覆盖了链接库的一个同名常量"

如果用户在工程中定义了与链接库同名的常量，库中该常量将被覆盖。

1970

"参数管理器：'<Name>' 列，'<Name>'行的'<Name>'值不能被引入!"

请检查数据引入文件*.prm 的格式是否符合当前参数管理器的设置。

1980

"对全局变量'<Name>'同时读、写可能导致数据丢失"

1990

" '<name>'参数未能在参数设置中获的可用的内存地址映射" 请在资源选项卡中的 参数设置窗口对其进行正确设置。

2500

"任务 '<任务名称>': 对循环任务没有指定循环时间"

10.42 错误

3100

"用户程序代码太大。系统支持代码最大容量为'<number>'字节(<number>K)" 超过最大程序规模。减小项目规模。

3101

"用户程序使用数据太多，最大容量为'<number>'字节(<number>K)"

超出内存容量范围。请减少应用程序的数据使用量。

3110

"'<Name>'库出现错误"

该库文件*.hex 不是 INTEL 16 进制格式。

3111

" '<Name>'库太大。系统支持最大容量 64K"

.hex 文件超过设定的最大容量。

3112

"库中存在不可重复定位指令"

该库代码无法进行链接。

3113

"库代码覆盖功能表格."代码和功能表的范围重叠。

3114

"库使用了多个内存段."

.hex 文件中的表和代码利用不止一个字段。

3115

"不能把常量赋值给 VAR_IN_OUT 型变量，数据类型不匹配."

由于数据设置在“near”(近的),但是串常数设置在“huge”（大的）或“far”（远的），因而串常数的内部指针格式不能转换成 VAR_IN_OUT 的内部指针格式。如有可能，改变这些目标设置。

3116

"功能表格覆盖库代码或段分界线。"

3117

"<Name> (<Zahl>):表达式过于复杂，无法用寄存器装载处理"

3120

"当前代码段超出 64K."

当前生成的代码大于 64K。最终建立太多的初始化代码。

3121

"POU 太大."POU 的大小应限制在 64K 以下。

3122

"功能或结构的初始化代码太大，超出 64K 上限"

用于一个功能或一个结构化 POU 的初始化代码不应超过 64K。

3123

"数据段太大：段'<Number>%s', 大小<size> 字节(最大为 <number>字节)"

3124

"字符串常量太大: '<number>' 字节 (最大 253 字节)"所给出的常量必须缩短字符数

3130

"用户堆栈太小：需要'<number>' 双字容量的堆栈，只能提供'<number>'双字容量的堆栈."

POU 调用的嵌套深度太大。在目标设置中输入一个更大 的堆栈尺寸 ，或编译没有任选项“debug”(可在对话框“project”(项目)“(options)”任选项“build”(建立)中设置)的建立项目。

3131

"用户堆栈太小：需要'<number>' 双字容量的堆栈，只能提供'<number>'双字容量的堆栈."

请联系 PLC 制造商。

3132

"系统堆栈太小：需要'<number>' 双字容量的堆栈，只能提供'<number>'双字容量的堆栈."

请联系 PLC 制造商。

3150

"功能 '<名称>'的参数 <编号>: 不能作为 C－函数的一个字符串参数来传递 IEC－功能的结果."

使用一个中间变量，将 IEC 功能的结果赋值给该中间变量。

3160

"不能打开'<name>'库文件."

工程通过库管理器包含了'<name>'库，但是在指定的路径未能找到该库文件。

3161

"库文件'<name>'缺少代码段"

一个程序库的一个.obj 文件至少应包含一个 C 功能。将一个亚功能插入到.lib 文件内，在.lib 文件内尚未定义这个亚功能。

3162

"无法解决库 '<名字>'涉及到的(符号 '<名称>', 类'<名称>', 类型'<名称>')" .obj 文件包含一个对其他符号不能分辨的引用。请检查 C 编译程序的设置。

3163

"'<name>'库包含未知的引用类型"

.obj 文件包含一个引用类型，它不能被代码发生器所分辨。请检查 C 编译程序的设置。

3200

"'<name>'：布尔表达式过于复杂，请使用中间变量简化之"

目标系统的暂时存储器容量对表达式的规模来说显得不够。将表达式分成几个部分表达式，为此可使用对几个中间变量进行赋值。

3201

"<名称> (<网络>): 一个网络必须不能导致代码超过 512 字节。"不能分辨内部跳转。启动选项“利用 16 位跳转补偿”中的 68K 目标设置。

3202

"功能嵌套调用导致堆栈溢出"

使用一嵌套的功能调用 CONCAT(x,f(i))。这会导致数据丢失。将调用分成两个表达式。

3203

"表达式过于复杂（无法使用寄存器进行处理）."

将赋值分成几个表达式。

3204

"程序跳转超出范围 32k 字节"

跳转范围应不超过 32767 字节。

3205

"内部错误：包含太多常量字符串"

在一个 POU 中，最多能使用 3000 个常量字符串。

3206

"功能块程序代码溢出"

功能块程序代码上限为 32Kb。

3207

"数组优化失败"

计算索引时功能被调用导致最优化数组存取失败。

3208

"转化未成功"

使用的转换功能，不是为实际代码发生器而实现的。

3209

"操作未成功"

使用的操作符，不是为这种数据类型和实际代码发生器而实现的。MIN(字符串 1，字符串 2)

3210

"'<Name>'功能未找到"

调用的功能不能用于项目内。

3211

"字符串使用超出范围"

字符串变量一个表达式中可最多使用 10 次。

3212

"在 POU <POU 名称>错误的库命令 "

3250

"8 位控制器不支持 REAL 类型"

目标当前未得到支持。

3251

" 8 位控制器不支持日期类型。"

目标当前未得到支持。

3252

"堆栈溢出<number>字节"

当前目标系统不支持。

3253

"找不到此 hex 文件：'<Name>'"

目标当前未得到支持。

3254

"调用的外部库功能不能被解析."

目标当前未得到支持。

3255

"8 位控制器不支持指针."

运行于 8 位系统上的程序应避免使用指针.

3260

"功能'<名称>' 拥有太多的自变量: 增加目标系统自变量堆的大小."

3400

"读取输入变量时发生错误"

.exp 文件包含一个正确的存取变量段。

3401

"装载变量配置时发生错误"

.exp 文件包含一个正确的配置变量段。

3402

"装载全局变量时发生错误"

.exp 文件包含一个正确的全局变量段。

3403

"目标<name>不能被装载"

.exp 文件中的对象<name>段不正确。

3404

"装载任务配置时发生错误"

.exp 文件中的任务配置段不正确。

3405

"装载 PLC 配置时发生错误"

.exp 文件中的 PLC 配置段不正确。

3406

"SFC 程序组织单元存在同名步"。第二步将不能载入。"

.exp 文件中的 SFC POU 段包含有相同名的两个步。在输出文件中重新命名其中的一步。

3407

"当前步的上一步未找到"

.exp 文件中丢失步<name>。

3408

"当前步的后续步未找到"

.exp 文件中丢失步<name>。

3409

" '<' name>'步转换条件不正确 "

.exp 文件中，丢失转换名，它需要步<name>作为先行步。

3410

″ 对应转换条件'<name>'的步不正确″ .exp 文件中，丢失一个步，它需要转换名<name>作为先行条件。

3411

″ '<name>'步与初始步的联系丢失″

.exp 文件中，丢失步<name>与初始步之间的连接。

3412

″宏'<name>'不能被载入″

3413

″CAM 文件载入时发生错误.″

3414

″CNC 文件载入时发生错误″

3415

″报警配置载入时发生错误″

3450

″'<PDO-name>':COB-ID 丢失″

为此模块，点出 PLC 配置对话框中的“properties”（属性），并输入用于 PDD<PDD 名>的一个 COB ID。

3451

″EDS 文件载入错误：硬件配置引用了此文件，但是找不到该文件″

CAN 配置所需的设备文件不在正确的目录内。检查“project”（项目） “options”（选项）“目录”中的配置文件的目录设置。

3452

″'<Name>'模块不能被创建!″

<Name>'模块的设备文件不兼容当前的设置，应根据 CoDeSys 的相关设置对其进行修改。

3453

″'<Name>'通道不能被创建!″

'<Name>'通道的设备文件不兼容当前的设置，应根据 CoDeSys 的相关设置对其进行修改。

3454

"'<name>'地址指向的内存已使用!"

在对话框 PLC 配置的“setting” （设置）中，已启动选项“check for overlapping addresses ” （检查重叠地址），并已检测到一个重叠。注意， 区域检查是基本尺寸，它与模块的数据类型有关。在配置文件中，通过登录项“size”（尺寸），给出区域尺寸。

3455

"载入 GSD 文件出错：硬件配置中要引用此文件，但是找不到该文件!"

Profibu 配置所需的设备文件不在正确的目录内。检查“project”（项目） “options”(选项) “目录”中的配置文件的目录设置。

3456

"总线设备驱动'<name>'无法被创建!"

用于模块<name>的设备文件不适应于当前的配置。自建立配置以来已作了修改，或者它已损坏。

3457

"模块解析错误!"

请检查这个模块的数据文件。

3458

"PLC 配置无法完成，请检查配置文件."

3459

"不支持所选波特率."

3460

3S_CanDrv.lib 库版本号错误。

3461

"3S_CanOpenMaster.lib 库版本号错误."

3462

"3S_CanOpenDevice.lib 库版本号错误."

3463

"3S_CanOpenManager.lib 库版本号错误."

3464

"3S_CanOpenNetVar.lib 库版本号错误."

3465

"CAN 通讯从模块的副索引应连续标号"

3466

"CAN 总线变量：PLC 配置中未发现 CAN 通讯控制器"

3468

"CAN 总线驱动：任务配置中未实现任务更新."

3469

"CANOpen 通讯主模块不能被调用，请手动分配其任务."

3470

"任务更新常数名称非法"

3500

"'<name>'变量未在变量列表中声明"将用于变量的说明插入到全局变量表内，该表包含“variable_configuration”（变量—配置）。

3501

"在变量列表中声明的'<name>'变量未分配相应的内存地址."将这个变量的一个地址分配给全局变量表，该表包含“variable_configuration”（变量—配置）。

3502

"变量配置列表中的变量'<name>'数据类型声明有误"

在包含“variable_configuration”（变量—配置）。的全局变量表中， 变量以不同于 POU 中的数据类型说明。

3503

"变量配置列表中的变量'<name>'数据类型有误"

在包含“variable_configuration”（变量—配置）。的全局变量表中， 变量以不同于 POU 中的数据类型说明。

3504

"变量配置中不支持初始赋值"“variable_configuration”（变量—配置）的一个变量是以地址和初始值说明的。但是，一个初始值只能为没有地址分配的输入变量而定义的。

3505

’<name>’路径不是合法的实例路径名 ″

“variable_configuration”（变量—配置）包括一个存在的变量。

3506

″存取路径未指定″在“access variables” （存取变量）的全局变量表中，一个变量的存取路径是不正确的。正确的应该是：《标示符》：《存取路径》：《类型》《存取方式》

3507

″存取变量对应的内存使用不合法″

3550

″标识符’<name>’重复定义″使用一个相同的名来定义两个任务。重新命名其中的一个。

3551

″任务’<name>’中至少应包含一个程序调用″插入一个程序调用或删除任务。

3552

″任务’<name>’使用的事件变量’<name>’未定义″在任务属性的“single”（单）区域对话框中设置的一个事件变量，在项目未经全局说明。利用其他变量或全局定义该变量。

3553

″任务’<name>’中的事件变量’<name>’必须是布尔类型变量″

在任务属性的“single”（单）区域对话框中，使用一个类型 BOOL 的变量作为事件变量。

3554

″任务入口 POU 必须是程序或者全局功能块实例″

在该字段中，输入“program call” (程序调用)一个功能或一个不定义的 POU。输入一个有效的程序名。

3555

″任务入口 POU 包含非法参数″

在该字段中，所用的”append program call”（附加程序）参数不符合程序 POU 的说明。

3556

″当前目标系统不支持此任务″

3557

"任务数超出最大范围"

3558

"任务'<name>'的优先权超出合法范围>'"

3559

"任务'<name>'：当前目标系统不支持任务间隔调用模式"

3560

"任务'<name>'：当前目标系统不支持任务 freewheeling 调用模式"

3561

"任务'<name>'：当前目标系统不支持任务触发调用模式"

3562

"任务'<name>'：当前目标系统不支持任务外部触发调用模式"

3563

"任务'<name>'的调用间隔设置超出合法范围"

3564

" 当前目标系统不支持任务'<name>'所设置的外部触发事件'<name>'"

3565

"定义的事件触发任务数超出最大范围"

3566

"定义的间隔运行任务数超出最大范围"

3567

"定义的 freewheeling 任务数超出最大范围"

3568

"定义的外部事件触发任务数超出最大范围"

3569

"设定为系统事件触发的POU 未定义"

3570

"多个任务设定的优先权相同"

3571

"工程未包含 SysLibCallback.lib 库，不能生成系统事件."

3572

"任务'<name>'的看门狗时间间隔设定超出合法范围"

3573

"看门狗时间设定超出合法范围"

3574

"事件变量对应的内存地址不能在事件中重复使用"

3575

"任务'<名称>': 循环时间应该是 '<数字>'的整倍数 ."

3600

"隐含变量未找到!"

使用命令"rebuild all"（重建所有）。如依然得到出错消息，请于 PLC 制造商联系。

3601

"'<name>'变量名称是系统保留用名，请更改"给出的变量在项目中说明，虽然它是保持用于代码发生器。重新命名该变量。

3610

" 不支持'<name>'"给出的属性不被编译系统的当前版本支持。

3611

"提供的编译目录名'<name>'非法"用于编译文件的"project" （项目） —"options" （选项） —"directories"（目录）中给出的目录是无效的。

3612

"超出系统所能处理最大 POU 数量，编译被中止."在项目中所用的 POU 和数据类型太多。在"target setting/memory layout"（目标设置/存储器布

局）中，修改 POU 的最大数。

3613

"工程编译取消"编译过程被用户删除。

3614

"工程必须包含一个名为 PLC_PRG 的主程序或者进行任务配置"

建立一个类型”program”（程序）的初始 POU，或建立一个任务配置。

3615

"<名称>(主要的程序) 必须有程序类型"

在项目中所用的一个初始 POU 不是” program”（程序）类型。

3616

"程序不能在外部库中执行"

应作为外部库保存的项目包含一个程序。当使用该库时，不能使用这个程序。

3617

"内存不足"

增加你的计算机的虚拟存储器的容量。

3618

"当前代码生成器不支持位存取!"

当前设置的目标系统的代码发生器，不支持对变量的位存取。

3619

"库文件'<name>'与其对应的目标文件版本不合!"

3620

"PLC_PRG 不能出现在库文件中"

3621

"无法对编译文件'<name>'进行写操作"

3622

"无法创建符号文件'<name>'"

3623

"无法对引导工程文件'<name>'进行写操作"

3624

"目标设置 <目标设置 1>=<设置值> 与 <目标设置 2>=<设置值> 不兼容"

在目标设置对话框内检查并纠正这些设置(资源标签)。如果这些设置不可见且不可在那里编辑，请联系 PLC 制造商。

3700

" 工程中存在与库中同名的 POU"

一个 POU 名已用于项目内，这一名早已用于一个库 POU。重新命名 POU。

3701

" 目标名不能与POU 同名"

使用命令“project”（项目）--“rename object”(重新命名对象)，重新命名“object organizer（对象组织程序）”中的 POU 或在说明窗内改变 POU 的名。在那里，POU 名必须放置在临近的关键字

PROGRAM，FUNCTION 或 FUNCTIONBLOCK 中的一个。

3702

"标识符声明过量"

最大为 100 个标示符可输入到一个变量说明内。

3703

"标识符'<name>'重复定义 "

要注意，在 POU 的说明部分中，只允许一个标示符有给定的名。

3704

"数据存在递归调用"

3705

"PLC_PRG 中不允许使用 VAR_IN_OUT 型变量"

3706

"标识符 '常量' 只允许在 'VAR', 'VAR_INPUT', 'VAR_EXTERNAL' 和'VAR_GLOBAL' 中应用。"

3720

"AT 关键字应紧接内存地址"

在关键字 AT 后面加一个有效地址，或修改关键字。

3721

"只有变量以及全局变量能定位于某个内存地址"

将说明放置在一个 VAR 或 VAR-GLOBAL 说明区。

3722

"只有布尔型变量允许使用位地址"

修改地址或修改分配地址的变量类型。

3726

"常量不能存储于指定的内存中"

这种类型的变量不能放置在给定的地址上。

3727

"该地址不允许定义数组"

3728

"非法的内存地址"

3729

"内存地址'<name>'中存储的数据类型非法 "

3740

"非法类型 "

在一个变量说明中，使用无效的数据类型。

" VAR_IN_OUT 型变量不允许赋初值."

在 VAR_IN_OUT 变量的说明部分，去除初始化。

3780

"期待'VAR', 'VAR_INPUT', 'VAR_OUTPUT' 或 'VAR_IN_OUT'等类型变量" POU 名后的第一行必须包含这些关键字中的一个关键字。

3781

"期待 END_VAR 标识符"

在说明窗内的给出行的起始位置输入一个 END—VAR 的有效标示符。

3782

"意外的结束"

在说明性编辑器中：在说明部分结束处加上关键字 END—VAR。在编译部分的文本编译器中：加上一个指令，它中止最后的指令顺序。

3783

"期待 END_STRUCT 标识符"

应确实弄清楚，类型说明已正确的终止。

3784

"当前目标系统不支持设定的此属性"

3800

"全局变量占用大量内存，请在工程选项增加内存使用量."

增加在对话框”project” （项目）--“options” （选项）--“build”（建立）的设定中所给出的程序段数。

3801

"'<name>'变量太大"

变量使用大于 1 个数据段的类型。段尺寸是一个目标特定的参数并可在目标设定/存储器布局中修改。如果你在当前的目标设定中找不到它，请与你的 PLC 制造商联系。

3802

"保留内存溢出."

可供保变量用的内存空间已耗尽。在目标设定存储器布局中可设定目标专用的内保存区域尺寸。如果你在对话框中找不到设定字段，请与你的 PLC 制造商联系。

3803

"全局数据存储内存溢出."

可供全局变量用的内存空间已耗尽，在目标设定/存储器布局中可设定目标专用的内保存区域尺寸。如果你在对话框中找不到设定的字段。请与你的 PLC 供应商联系。

3820

"功能中不允许使用 VAR_OUTPUT 和 VAR_IN_OUT 型变量"

在一个功能中，可以定义无输出或输入-输出变量。

3821

"功能至少应有一个输入接口"

为这种功能至少附加一个输入参数。

3840

"未知的全局变量'<name>'!"

在 POU 中，使用一个 VAR_EXTERNAL 变量，对它未说明其全局变量。

3741

"非法的数据类型"

使用一个关键字或一个操作符以代替一个有效类型标示符。

3742

"应指定枚举值"

在枚举类型定义中，在打开的括号后，或在括号之间的逗号后面，丢失一个标示符。

3743

"枚举应使用整型数字进行初始化"

枚举只能以类型 INT 的数初始化。

3744

"枚举常量名已被定义过"

检查一下，你是否遵循了以下的枚举值定义规则： .在一个枚举定义中，所有的值都应是唯一的。

.在所有的全局枚举定义中，所有的值均应是唯一的。

.在所有的局部枚举定义中，所有的值均应是唯一的。

3745

"子区域中只允许使用整型数据类型!"

子范围类型只能在整数数据类型上定义。

3746

"子区域'<name>'与基本数据类型不兼容"

子范围类型的范围极限设定中，有一个极限设定超出其基本类型的有效范围

3747

"字符串'<name>'长度未知"

没有一个有效的常数用于串长的定义。

3748

"最多只能定义 3 维数组"

在一个数据的定义中，给出允许的维数即大于三维。若可应用的话，使用“ARRAY OF ARRAY”（数组的数组）。

3749

"下限'<name>'未定义"

使用一个为定义的常数来定义一个子范围或数组类型的下限。

3750

"上限'<name>'未定义"

使用一个为定义的常数来定义一个子范围或数组类型的上限。

3751

"字符串长度非法"

3752

"数组嵌套使用最大不能超过9 维"

3760

"初始值错误"

使用一个与类型定义相当的初始值。为了更改说明，你可使用变量的说明对话框。（SHIFT F2 或 EDIT

（编辑）“autodeclare”(自动说明)）。

3841

"'<name>'声明与全局声明不匹配!"

在 VAR_EXTERNAL 变量说明中给出的类型与全局说明中的类型不匹配。

3850

"在包装的'<名称>'结构体内部对未包装'<名称>'的结构体的声明是不允许的!"

3900

"标识符的多重强调"

在标示符名下除去多重下划线。

3901

"地址表达式至多允许包含 4 个数字段"

有一种表达式允许直接赋值给一个地址，它有不止四层的数值域。

3902

"关键字必须大写"

对关键字应使用大写字母，或启动选项，“project” （项目） —“options”--（选项）中的“autoformat”（自动格式化）。

3903

"非法的 duration 型常量"

常数的计数法不符合 IEC61131-3 格式。

3904

"duration 型常量溢出"

用于时间常数值不能以内部格式表示。可表示的最大值为 t#49d17h2m47s295ms

3905

"非法的日期常量"

常数的计数法不符合 IEC61131-3 格式。

3906

"非法的时间常量"

常数的计数法不符合 IEC61131-3 格式。

3907

"非法的日期和时间常量"

常数的计数法不符合 IEC61131-3 格式。

3908

"非法的字符串常量"

串常数包含一个无效的字符。

4000

"期望标识符"

在这个位置输入一个有效的标示符。

4001

"没有声明变量'<名称>'"

说明变量是局部或全局。

4010

"类型搭配错误：不能转换'<名称>'为'<名称>'."

检查一下，操作符期望的是何种数据类型，并改变引起错误的变量类型，或选择其他变量。

4011

"在'<名称>'中参数类型搭配错误：不能转换'<名称>'为'<名称>'."

实际参数的数据类型不能自动转换成格式化参数的类型。使用某种类型转换方式或使用其他变量类型。

4012

"在'<名称>'中参数类型搭配错误：不能转换'<名称>'为'<名称>'."

将一个无效类型《类型 2》变量分配给输入变量'<name>'。将变量或常数以一个类型《类型 1》变量代之，或采用一种类型转换。相应的，一个带有类型前缀的常数。

4013

"在'<名称>'中输出类型搭配错误：不能转换'<名称>'为'<名称>'."

将一个无效类型《类型 2》变量分配给输入变量'<name>'。将变量或常数以一个类型《类型 1》变量代之，或采用一种类型转换。相应的，一个带有类型前缀的常数。

4014

"无夸张类型：不能转换'<名称>'为'<名称>'"

常数类型与前缀类型不兼容。

4015

"对于直接的位存储数据类型'<名称>'不合法"

直接位编址只允许用于整数和"位串"数据类型。你在位存取<var1>.<bit>中，使用 RIAL/LREAL 类型的变量 var1,或一个常数。

4016

″对于’<名称>’类型的变量的位索引’<编号>’超出范围″

你正式试图存取的一位，它不是为变量的数据类型而规定的。

4017

″对于’REAL’，　’MOD’没有定义″

操作符 MOD 只可用于整数和位串数据类型。

4020

″具有写权的变量或直接地址需要’ST’,’STN’,’S’,’R’″

使用一个具有写存取的变量取代第一个操作数。

4021

″变量’<名称>’不具有写权″

使用一个具有写存取的变量取代该变量。

4022

″期望操作数″

在注释后面加上一个操作数。

4023

″在’＋’或’－’后需要数字″

输入一个数字。

4024

″在’<名称>’前需要<操作数 0>或<操作数 1>’或……″

在命名的位置处，输入一个有效的操作数。

4025

″在’<名称>’前需要’:=’或’=>’″在命名的位置处，输入二种操作符中的一种。

4026

″‘BITADR’需要一个位地址或关于位地址的一个变量″

使用一个有效的位地址。

4027

″需要一个整数符号或常数符号″

输入一个整数或一个有效常数的标示符。

4028

″‘INI’操作数需要功能块实例或数据单元类型实例″

检查 INI 操作符所使用的变量的数据类型。

4029

″不允许对同一个功能进行嵌套.″

在不是可重入的目标系统和在仿方式时，一个功能调用并不包含作为参数的自身调用。

4030

″‘ADR’操作数不允许用表达式或常量″

使用一个变量或一个直接地址取代常数或表达式。

4031

″位操作不允许用’ADR’！请用‘BITADR’来代替.″

使用 BITADR。请注：BITADR 功能不返回一个物理的内存地址。

4032

″对于’<名称>’来说，’<数字>’操作数太少。至少需要’<数字>’个″

检查一下，命名的操作符需要多少操作数，并添加缺少的操作数。

4033

″对于’<名称>’来说，’<数字>’操作数太多。至少需要’<数字>’个″检查一下，命名的操作符需要多少操作数，并除去多余的操作数。

4034

″用 0 来除″

你正在一个常数表达式中以 0 作为除数。如果你要引起一个运行时的错误，使用—如可以应用的话—一个有值为 0 的变量。

4035

″如果’代替常量’被激活，’VAR’必须在’VAR CONSTANT’中应用″

使用直接值 的一个常数地址存取是不可能的。 如可应用的话，取消”project”（项目）---“options”（选项）---“build”（建立）

4040

"标签'<名称>'没有定义"

使用名（标记名）定义一个标记，或将名（标记名）改变成一个定义的标记名。

4041

"标签'<名称>'进行了多重定义"

在 POU 中，标记<name>是重复定义的。重新命名标记或从重复定义中除去一个名。

4042

"在序列中没有比'<数字>'更多的标签可以被允许"

跳转标记数限制在"<Anzahl>"。插入一个哑指令。

4043

"标签格式错误。一个标签在冒号后必须定义一个随意的名字."

标记名是无效的，或在表达式中丢失了冒号。

4050

"没有定义 POU'%S'"

利用命令"project"（项目）--"Add Object"（附加对象）来定义有名<name>的一个 POU，或将<name>改变成一个已定义的 POU 名。

4051

"'%S'没有功能"

替代<name>，使用一个在项目或在库中定义的功能名。

4052

"'<名称>'必须是 FB'<名称>'声明的一个实例"

使用一个在项目中定义的数据类型<name>的实例，或将（实例名）类型改为<name>。

4053

"'<名称>'没有有效的盒子或操作符"

使用一个 POU 名或一个在项目定义的操作符来取代<name>。

4054

"所给参数没有有效的 POU 名称"

给出的参数不是一个有效的 POU 名。

4060

″’<名称>’的‘VAR_IN_OUT‘参数’<名称>’需要写权作为输入″

具有写存取的变量必须移交给 VAR_IN_OUT 参数，这是因为，一个 VAR_IN_OUT 可在 POU 内修改。

4061

″’<名称>’的‘VAR_IN_OUT‘参数’<名称>’必须被用.″

VAR_IN_OUT 参数必须有移交的具有写存取的变量，这是因为，一个 VAR_IN_OUT 可在 POU 中修改。

4062

″’<名称>’的‘VAR_IN_OUT‘参数’<名称>’没有外部接口.″

VAR_IN_OUT 参数只能在 POU 内写入或读取，这是因为它们是由引用转交的。

4063

″’<名称>’的‘VAR_IN_OUT‘参数’<名称>’不准应用位地址.″

一个位地址不是一个有效的物理地址。转交一个变量或一个直接的非位地址。

4064

″在本地调用时‘VAR_IN_OUT‘必须复写!″

在局部动作调用中，删除用于 VAR_IN_OUT 变量的参数集。

4070

″POU 包含太复杂的表达式″

通过将表达式分成几个表达式来降代嵌套深度。为此，可使用中间变量。

4071

″网络太复杂″

将网络分成几个网络。

4072

″在 FB 类型’<名称>’和实例’<名称>’的动作标识符应用不一致.″

4100

‘^‘需要指针类型″

你正在试图将一个不作为指示符说明的变量非关联化。

4110

"[<索引>]需要数组变量"

[<index>]用于一个变量，它不是作为有 ARRAY OF 的一个数组加以说明的。

4111

"一个数组的索引表达式必须是‘INT‘类型"

使用正确类型的表达式，或类型转换。

4112

"数组有太多索引"

检查索引数（1.2 或.3），这些索引是说明数组的，除去多余的索引。

4113

"数组有太少索引"

检查索引数（1.2 或.3），这些索引是说明数组的，添加缺少的索引。

4114

"一个常量索引超出数组范围"

确实弄清楚，所用的索引是在数组界限之内。

4120

"‘.‘需要变量结构体"

句点左边的标识符必须是一个类型 STRUCT 或 FUNCTION_BLOCK 的变量，或是名为一个 FUNCTION 或一

个 PROGRAM 的变量。

4121

"’<名称>’不是’<对象名称>’的成分"

成分<name>不包括在对象<object name>的定义内。

4122

"’<名称> ’不是调用的功能块的输入变量"

检查一下，被调用的功能块的输入变量，并将’<name>’改变成这些变量中的一个变量。

4200

"需要 LD"

在 IL 编辑器中，在跳转符号后面，至少应插入一个 LD 指令。

4201

"需要 IL 操作符"

每个 IL 指令必须用一个操作符或一个跳转符开始。

4202

"括号内文本意外结束"

在文本后插入一个关闭括号。

4203

"'<名称>'在括号内不允许"

操作符<name>在一个 IL 括号表达式中是无效的。（无效的是：JMP,RET,CAL,LDN,LD,TIME）

4204

"对于结束括号缺少相应的开始括号"

插入一个打开括号，或除去这个关闭括号。

4205

"在‘）‘后不允许逗号"

在关闭括号后面除去逗号。

4206

"括号内不允许有标签"

移去跳转符号，使其在括号外面。

4207

"‘N‘修正符需要操作数类型’BOOL‘’BYTE‘’WORD‘或’DWORD‘"

N 修改符需要可执行布尔求反的数据类型。

4208

"条件操作符需要‘BOOL‘类型"

确实弄清楚，表达式给出布尔结果，或应用类型转换。

4209

"在此不允许功能名"

使用一个变量或一个常数来替换这个功能。

4210

"'CAL', 'CALC' 和 'CALN'需要功能块实例作为操作数"

说明一个你要调用的功能块的实例。

4211

"在 IL 程序中只允许在行的末尾进行注释"

将注释移到行的结束处，或移到附加行。

4212

"条件声明前的累加器错误"

未定义这个累加器。如果一个指令是领先的，它未提交一个结果。

4213

"'S'和'R'需要'BOOL'操作数"

在这个位置使用一个布尔变量。

4250

"在 POU 结尾需要另一个'ST'声明"

该行不是以一个有效的 ST 指令开始。

4251

"在功能'<名称>'内有太多参数"

给出的参数多于功能定义中所说明的参数。

4252

"在功能'<名称>'内有太少参数"

给出的参数少于功能定义中所说明的参数。

4253

"'IF'或'ELSIF'需要'BOOL'表达式作为条件"

确实弄清楚，IF 或 ELSIF 的条件是一个布尔表达式。

4254

″’ WHILE‘需要’BOOL‘表达式作为条件″

确实弄清楚，WHILE 后的条件是一个布尔表达式。

4255-

″’ UNTIL‘需要’BOOL‘表达式作为条件″

确实弄清楚，UNTIL 后的条件是一个布尔表达式。

4256

″‘NOT’ 需要’BOOL‘操作数″

确实弄清楚，NOT 后的条件是一个布尔表达式。

4257

″‘FOR‘声明的变量必须是’INT‘类型″

确实弄清楚，计数器变量是一个整数或位串数据类型。（例如：DINT，DWORD）

4258

″‘FOR‘声明的表达式没有具有可写权的变量″

使用一个有写存取的变量来替换计数器变量。

4259

″‘FOR‘声明的开始值没有具有可写权的变量″

“FOR”指令中的起始值必须与计数器变量类型相兼容。

4260

″‘FOR‘声明的结束值没有具有可写权的变量″

“FOR”指令中的结束值必须与计数器变量类型相兼容。

4261

″‘FOR‘声明的增加值没有具有可写权的变量″

FOR”指令中的起始值必须与计数器变量类型相兼容。

4262

″循环外需要退出″

只能在“FOR”，“WHILE”或“UNTIL”指令内使用“EXIT”。

4263

″需要数字，‘ELSE‘或‘END_CASE‘″

在一个“CASE”表达式内，你只能使用一个数或一个“ELSE”指令，或结束指令“END_CASE”。

4264

″‘CASE‘需要整数类型选择器″

确实弄清楚，选择器是一个整数或位串数据类型的（例如 DINT，DWORD）。

4265

″‘，‘后需要编号″

在 CASE 选择器的枚举时，在逗号后面必须插入一个其他选择器。

4266

″至少需要一个声明″

插入一个指令，至少一个分号。

4267

″功能块的调用需要功能块实例″

功能块的调用中的标识符没有实例。说明所需要的功能块的一个实例，或利用一个早已定义的实例。

4268

″需要表达式″

在这里插入一个表达式。

4269

″在'ELSE'分支后需要有'END_CASE'″

使用一个“END_CASE”来终止“ELSE”部分后面的“CASE”指令。

4270

″'CASE' 常量'<名称>' 已经被应用″

一个“CASE”选择器在一个“CASE”指令内只能使用一次。

4271

″范围的下限值比上限值大.″

修改选择器的区域边界，使其下部边界不大于上部边界。

4272

"在调用'<名称>'的地方<位置>需要参数'<名称>'!"

你可以这样的方式编辑一个功能调用，使它还包含参数名，而不仅包含参数值。但是，无论如何，参数的位置（顺序）必须与功能定义中相同。

4273

"部分'CASE'范围'<范围>'在'范围'<范围>'中已经被应用"

确实弄清楚，用于 CASE 指令中的选择器区域不出现重叠。

4274

"在'CASE'声明时出现多重的 'ELSE'分支"

一个 CASE 指令不能包含多于一个 ELSE 指令。

4300

"跳转需要'BOOL'作为输入类型"

确实弄清楚，相应于 RETURN 指令的跳转输入是一个布尔表达式。

4301

"POU '<名称>' need exactly 需要正确的 <数字> 输出"

输入数不对应于在 POU 定义中给出的 VAR_INPUT 和 VAR_IN_OUT 变量数。

4302

"POU '<名称>' 需要正确的输出 %d ".

输出数不对应于在 POU 定义中给出的 VAR_OUTPUT 变量数。

4303

"'<名称>' 没有操作符"

使用一个有效的操作符来替换<name>。

4320

"一个触点的开关信号必须是布尔类型的表达式"

用于一个接点的开关信号必须是一个布尔表达式。

4321

"线圈的输出变量必须是布尔类型。"

一个线圈的输出变量必须是类型 BOOL。

4330

"功能块 '<名称>'的输入'EN'处期望一个表达式 "

将一个输入或一个表达式分配给 POU "<name> "的输入 EN。

4331

"功能块 '<名称>'的输入'<编号>'处期望一个表达式 " 没有分配操作符 POU 的输入<number>。

4332

" 功能块 '<名称>'的输入'<名称>'处期望一个表达式"

POU 的输入是类型 VAR_IN_OUT 而且没有分配。

4333

"跳转处期望一个标识符"

给出的跳转标记不是一个有效的标识符。

4334

"跳转的输入处期望一个表达式"

将一个布尔表达式分配给 RETURN 指令的输入。如这是 TRUE，则将执行这个跳转。

4335

"返回的输入处期望一个表达式"

将一个布尔表达式分配给 RETURN 指令的输入。如这是 TRUE，则将执行着个跳转。

4336

"输出的输入处期望一个表达式"

将一个合适的表达式分配给输出框。

4337

"期望输入的标识符"

在输入框中插入一个有效的表达式或标识符。

4338

" '<名称>'功能块没有输入"

对操作符 POU<name>输入分配有效的表达式。

4339

"输出时的类型搭配错误: 不能转换 '<名称>' 到 '<名称>'.

输出框中的表达式类型与应分配给它的表达式类型不兼容。

4340

"跳转需要 'BOOL' 作为输入类型"

确实弄清楚，跳转是一个布尔表达式。

4341

"返回需要一个布尔输入"

确实弄清楚，用于 RETURN 指令的输入是一个布尔表达式。

4342

"在功能块 '<名称>'的输入'EN'处需要一个表达式"

将一个有效的布尔表达式分配给框的 EN 输入。

4343

"常量的值和声明的不一致"

框<name>的输入<name>作为 VAR_INPUT CONSTANT 说明。但是， 在对话框“编辑参数”中，已将一个

类型不兼容的表确定式发配给这个 POU 框。

4344

"'S' 和'R' 需要'BOOL' 操作数"

在相应的“复位”指令的设定后面，插入一个有效的布尔表达式。

4345

" '<名称>'的无效的参数类型'<名称>' : 不能转换 '<类型>' 到'<类型>'." 分配给 POU 框<name>的输入<name>的一个表达式。类型是不兼容。

4346

"不允许一个常量作为输出"

你只能将一个输出分配给一个变量或一个有写存取的直接地址。

4347

"'VAR_IN_OUT' 参数需要可写的输入变量"

只有写存取的变量才可移交给 VAR_IN_OUT 参数，这是因为这些变量可在 POU 内修改。

4348

"无效的程序名称 '<名称>'. 具有同样名称的一个变量已经存在."

4349

"在 POU <名称>中的输入或输出已经被删除: 检查功能块的所有连接.只有当 CFC 编辑后此错误消息才会消失。"

4350

"不能从外部访问一个 SFC-行为动作!"

SFC 动作只能在它们被定义的 SFC POU 内才能调用。

4351

"步的名称没有标识符: '<名称>'"

重新命令步名，或选择一个有效的标识符作为步名。

4352

"有效的步名称后有额外的字符:'<名称>'"

在步名中除去无效的字符。

4353

"步名称被复写: '<名称>'"

重新命名一个步名。

4354

"跳转到没有定义的步: '<名称>'"

选择一个现有的步名作为相应的跳转的目标。插入一个有名的步，“<name>”

4355

"一个转换必须没有任何的边缘效应(分配, FB-调用等.)"

一次转换必须是一个布尔表达式。

4356

"跳转时没有有效的步名称: '<名称>' "

使用一个有效的标识符作为跳转目标（标记）。

4357

"没有找到 IEC-库"

检查一下，库 iecsfc.lib 是否插入在库管理程序内，以及在”project”（项目）——“options”（选项）——“paths”(路径)中规定的库路径是否正确。

4358

"行为动作没有声明: '<名称>'"

确实弄清楚，在对象组织程序内，IEC 步的动作是在 SFC POU 下面插入的，而且在编辑器中，动作

名插入在限定符右边的框内。

4359

"无效的限定词: '<名称>'"

在动作名左边的框内，输入一个用于 IEC 动作的限定符。

4360

" 限定词'<名称>'后期望时间常数"

紧邻动作名左侧的框，在一个限定符后输入一个时间常数。

4361

"'<名称>' 不是行为动作的名称"

紧邻限定线右侧的框，输入一个动作名，或在项目中定义的一个变量名。

4362

"行为动作中没有布尔表达式的应用: '<名称>'"

插入一个布尔变量或一个有效的动作名。

4363

"IEC-步名称已经被变量应用: '<名称>'"

请重新命名步或变量。

4364

"一个转换必须是布尔量表达式"

转换表达式的结果必须是类型 BOOL。

4365

"限定词'<名称>'后期望有时间常数"

打开步"<name>的对话框"步属性"，并输入一个有效的时间变量或时间常数。

4366

"平行分支的标签没有有效的标识符: '<名称>'"

在三角形符号旁输入一个有效的标识符，它标记跳转符。

4367

" '<名称>' 标签已经用过"

早已有一个跳转符或一个步使用过这个名。请相应的重新命名。

4368

"'<名称>'行为在多重的一系列步中应用，也就是一个包含其它的。"

动作<name>用于 POU 以及 POU 的一个或几个动作。

4369

"一个转换需要一个严密的网络"

对于一个转换使用了几个 FBD 相应的 LD 网络。请减少到一个网络。

4370

"正确的 IL-转换后有额外的行"

在转换结束处除去不必要的线路。

4371

"有效的表达式后有无效的字符: '<名称>"

在转换结束处除去不必要的字符。

4372

"'<名称>'步: 时间限制需要'TIME'类型"

4373

" 只在 SFC-POUs 下允许 IEC-行为"

4374

"期望步代替'<名称>'转换 "

4375

"期望转换代替'<name>'步"

4376

"'<名称>'转换后期望步"

4377

" '<名称>'步后期望转换"

4400

POU '<名称>'输入/ 转换 有错误。例如没有完成."POU 不能完全转换到 IEC61131-3，相应的丢失一个操作数。

4401

"S5 时间常量 <名称> 秒数太大 (最大. 9990s)."

在累加器内没有有效的 BCD 编码时间。

4402

"只有 I/O 可进行直接访问."

确实弄清楚，你只存取定义为输入或输出的变量。

4403

"STEP5/7 结构体无效或不能转换成 IEC 61131-3."

有些 STEP5/7 命令不能转换到 IEC61131-3，相应的丢失一个操作数。

4404

"STEP5/7 操作数无效或不能转换成 IEC 61131-3."

有些 STEP5/7 操作数不能转换到 IEC61131-3，相应的丢失一个操作数。

4405

"重置一个 STEP5/7 定时器不能被转换成 IEC 61131-3."

相应的 IEC 定时器没有复位输入。

4406

"STEP5/7 计数器常量超出范围 (最大是 999)."

累加器中没有有效的 BCD 编码的计数器常数。

4407

"STEP5 结构体不能转换成 IEC 61131-3."

有些 STEP5/7 指令不能转换成 IEC61131-3，例如 DUF。

4408

"定时器 或 计数器 的位访问不能转换成 IEC 61131-3."

专用的定时器/计数器命令不能转换成 IEC61131-3。

4409

"ACCU1 或 ACCU2 的内容没有定义，不能转换成 IEC 61131-3."

与两个累加器相连接的一个命令是不能转换的，这是因为未规定累加器的值。

4410

"没有在工程中调用 POU ."

输入被调用的 POU。

4411

"全局变量表出现错误."

请检查 SEQ 文件。

4412

"内部错误编号 11"

请与 PLC 制造商联系。

4413

"在数据块中格式化行时出错"

应输入的代码中有一个错误的日期。

4414

"FB/FX 名称丢失."

在需始的 S5D 文件中，丢失一个（扩展的）POU 的符号名。

4415

"不允许块结束后的说明."

不能输入一个受保护的 POU。

4416

"命令无效"

S5/S7 命令不能被取消。

4417

"注释没有封闭"

使用“*）”来关闭注释。

4418

"FB/FX-名称太长 (最多 8 个字符)"

一个（扩展的）POU 的符号名太长。

4419

"期望对行""(* 名称: <FB/FX-名称> *)""格式化 "

校正相应的行。

4420

"FB/FX 参数名称丢失"

检查 POU。

4421

"FB/FX 参数类型无效"

检查 POU。

4422

"FB/FX 参数类型丢失"

检查 POU。

4423

" FB/FX 调用参数无效。"

检查 POU 的接口。

4424

"警告: 调用 FB/FX 时 丢失或参数错误或有'0'参数。"

被调用的 POU 尚未输入，或不正确，或无参数（在最后一种场合，你可忽略出错消息）。

4425

"标签定义丢失"

未定义跳转的目标（标记）。

4426

"POU 没有有效的 STEP 5 块名称, 例如 PB10"

修改 POU 名。

4427

"时间类型没有声明"

在全局变量表中加入一个定时器的说明。

4428

"超出打开 STEP5 支架的最大数量。"

不允许使用多于七个的开括号。

4429

"正式参数的名称错误。"

参数名不能超过四个字符。

4430

"参数的正式类型不是 IEC 可改变的。"

在 IEC61131-3 中，定时器，计数器和 POU 不能转换为形式参数。

4431

"调用 STEP5 STL 时其 'VAR_OUTPUT' 有太多的参数。"

一个 POU 不能包含多于 16 个形式参数作为输出。

4432

"不允许标签带表达式。"

在 IEC61131-3 中，跳转标记不能插入在任何需要的位置。

4434

"太多标签。"

一个 POU 不能包含多于 100 个标记。

4435

"在转移 / 调用后，必须启动一个新的表达式"

跳转或调用后，必须后随一个“load”（装入）命令 LD。

4436

"结果位没有定义，不能转换成 IEC 61131-3."

VKE 使用的命令不能转换，这是因为，VKE 的值是未知的。

4437

"指令和操作数的类型不一致"

一个值命令用于一个字操作数（或反过来也如此）。

4438

"数据块没有打开 (在 DB 之前插入 C 指令)"

插入一个 ADB。

4500

"变量或地址未被承认"

在项目内未说明监视变量。通过按 F2 按钮，你得到列出说明变量的输入辅助。

4501

"有效监视表达式后的额外特性。"除去多于的记号。

4520

"程序错误： 在'<名称>'前应该有标记!"

附注不正确。检查一下，”<name>”是否是一个有效的标记。

4521

"程序错误: 不期望的成分 '<名称>'!"

检查一下，附注是否正确编写。

4522

"标记超出程序的要求!"

丢失断开的附注，添加一个“flag off”（除去标记）指令。

4523

"程序{<程序名称>} 不允许的界面类型 '<名称>'"

4550

"索引超出定义范围 : 变量 OD "<编号>, 行<行号>."

保证索引是在目标设定/网络功能度中所规定的区域内。

4551

"分索引超出定义范围 : 变量 OD "<编号>, 行<行号>."

保证子索引是在目标设定/网络功能度中所规定的区域内。

4552

"索引超出定义范围 : 参数 OD "<编号>, 行<行号>."

保证索引是在目标设定/网络功能度中所规定的区域内。

4553

"分索引超出定义范围 : 参数 OD "<编号>, 行<行号>."

保证子索引是在目标设定/网络功能度中所规定的区域内。

4554

"变量名称错误: 变量 OD <编号>, 行<行号>."

保证在字段”variable”（变量）中的一个有效项目变量。使用相应于全局变量(variable name) 的语法 POU 名（变量名）。

4555

"登陆表为空, 输入不能随意: 参数 OD <编号>, 行<行号>"

你必须在这个字段做出一个登录项。

4556

"登陆表为空, 输入不能随意: 变量 OD <编号>, 行<行号>"

你必须在这个字段做出一个登录项。

4557

"必须的参数内存太大"

4558

"必须的变量内存太大"

4560

"错误的值：路径'<名称>'，纵列'<名称>'，行'<行号>'"

4561

"纵列没有定义：'<名称>'"

4562

"索引/分索引 已经存在：路径'<名称>'，行'<行号>'"

4563

"标识符'<名称>' 已经存在：路径'<名称>'，行'<行号>'"

4564

"索引'<名称>' 超出范围：路径'<名称>'，行'<行号>' "在目标设置定义范围内输入一个索引。

4565

"分索引'<名称>' 超出范围：路径'<名称>'，行'<行号>' "

在目标设置定义范围内输入一个分索引。

4566

"参数管理器输入时发生错误"

用户应该输入一个在参数管理器中包含错误信息的输出文件。检查*.exp 文件。

4600

"网络变量：'<名称>' 表达式不是布尔类型。"

4601

"网络变量 '<名称>'：发现网络变量交换没有循环或无限循环。"

4602

"'<网络变量名称表>'：对象用 UDP 端口'<端口号>' 代替 '<端口号>'"

在指定的网络变量表 设置中，一个端口号已经应用，它和在全局变量文件夹中找到的第一个网络变量表应用的端口号不同。当心所有的网络变量表应用同一个端口！

4620

工程中发现没有使用的变量。请参考 ’工程’ ’检查’ 未使用变量 命令的描述。

4621

变量分配内存空间时发生交迭。请参考 ’工程’ ’检查’ ’重叠内存范围’命令的描述。

4622

IEC地址分配的同一个内存空间涉及到多个任务。请参考’工程’ ’检查’ ’同时访问’命令的描述。

4623

工程在多于一个地方同时获得对同一个内存的写入权。请参考 ’工程’ ’检查’’ ’输出量的多重访问’命令的描述。

4650

″轴组’<名称>’: 任务’<名称>’ 不存在.″

4651

″轴组’<名称>’: 没有设置循环时间(dwCycle).″

4652

″驱动器’<名称>’: wDriveID 在此轴组中已经存在.″

4670

″CNC 程序’<名称>’: 没有找到全局变量 ’<名称>’ .″

4671

″CNC 程序’<名称>’: 变量’<名称>’ 具有矛盾的类型.″

4685

″凸轮’<名称>’:未知的凸轮表类型.″

4686

″凸轮’<名称>’: 凸轮上的点超出了数据类型范围.″

4700

″’<编号>’ (’<名称>’): 监视表达式 ’<名称>’ 不是数字变量.″

4701

″’<名称>’ (’<编号>’): 监视表达式 ’<名称>’ 不是布尔类型.″

4702

″'<名称>' ('<编号>'): 监视表达式 '<名称>' 不是字符串类型.″

4703

″'<名称>' ('<编号>'): 监视表达式'<名称>'错误″

可视化窗口包含了错误的变量。

4704

″'<名称>'('<编号>'): 监视表'<名称>'初始化值错误.″

检查所用的表。

4705

″'<名称>' ('<编号>'): 报警表分配的报警组无效.″

在报警表配置对话框中输入一个有效的报警组。（报警种类表）

4900

″类型转换错误″

当前选择的代码转换器不支持你所用的类型转换。

4901

″内部错误：数组存取溢出!″

数组范围超过 32 位变量，减少数组索引范围。

5100

″<名称> (<Zahl>): 表达式太复杂。没有寄存器可用。″

对于可用寄存器来说指定的表达式太复杂。请尝试用中间变量来减少表达式的复杂度。