Twido
可编程控制器
软件手册

目录

重要信息

注意　请参照设备仔细阅读说明书，以便在安装，操作和维护之前熟悉设备。以下特殊信息会在本手册和设备中出现，用以警告潜在的危险。

当这个符号和 Danger 或 Warning 安全一起出现时，意味着存在电气危险，如果不按照说明书操作，将会造成人身伤害。

这是一个安全警告符号，警告用户存在潜在的人身伤害。请遵循这个符号后的所有安全信息，以避免伤害或死亡。

⚠ DANGER
DANGER 意味着即将发生危险，如果不避免这种情况，必然会引起死亡，严重伤害或设备损坏。

⚠ AVERTISSEMENT
WARNING 意味着存在潜在危险，如果不避免这种情况，将会引起死亡，严重伤害或设备损坏。

⚠ ATTENTION
CAUTION 意味着存在潜在危险，如果不避免这种情况，将会引起伤害或设备损坏。

注意

只有合格用户才能使用电气设备。Schneider Electric不负担任何违反本手册操作而引起的责任。本手册仅限于受过培训的用户使用。在Twido硬件手册中提供了装配和安装的指导。

Reference Manual, TWD USE 10AF.

附加安全信息

每一个对使用和安装本产品负责的用户必须确定每一个应用程序已经包含所有必要的设计考虑，并且符合所有的应用规则，性能与安全要求，规章，编码和标准。

一般性警告和注意

WARNING

爆炸危险

- 替代元件可能会损坏Class I, Div 2适应性。
- 在带电或危险区域中，请不要断开设备的连接。

如果不注意这个警告，会导致严重伤害或设备损坏。

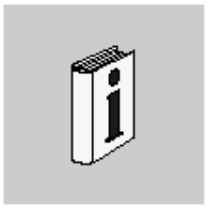
WARNING

无意识的设备操作

- 在安装，搬动，接线和维护之前请关闭电源。
- 本产品不适用于临界安全环境。在该环境下，人员或设备的危险依然存在，需要使用适当的硬连线安全锁紧装置。
- 请不要拆开，修理或更改模块。
- 请在机柜中使用控制器。
- 请在规定的操作环境下使用模块。
- 请在传感器与模块相连并需要供电的情况下使用传感器电源供应。
- 请在电源线和输出电路上使用IEC60127认证的熔断器，以满足电压和电流的要求。建议使用：5x20 mm型218000系列/T型的慢熔型熔断器。

如果不注意这个警告，会导致严重伤害或设备损坏。

关于本手册



概览

手册范围

本软件手册针对Twido可编程控制器，主要包括以下几个部分：

- Twido编程软件描述及Twido控制器编程基本原理介绍。
- 通信，模拟I/O管理，AS-I总线接口模块安装和其它特殊函数描述。
- 用于创建Twido程序的软件语言描述。
- Twido控制器指令和函数描述。

有效性注释

本手册中的信息只适用于 Twido 可编程控制器。

版本信息

版本号	变化
3	源代码改为法语且更新

相关文档

与产品相关的警告

施耐德电气公司不对本手册中的任何错误负责。在事先未得到施耐德电气公司同意的情况下，本手册中的任何部分不允许以包括电子文档在内的任何形式或方式复制。

用户意见

我们欢迎您对本手册提出意见。您能通过e-mail：TECHCOMM@modicon.com联系我们。

概览

本部分的主题

本部分介绍了 Twido 可编程控制器的软件语言及创建控制程序所需要的基本信息。

本部分包含了哪些内容?

本部分包含了以下章节:

Twido 软件介绍

1

概览

本章的主题

本章简要介绍了 Twido 控制器的编程及配置语言：TwidoSoft，及 List，Ladder 和 Grafcet 编程语言。

本章包含了哪些内容？

本章包含了以下主题：

导言

TwidoSoft 是一个图形开发环境，用于 Twido 可编程控制器应用程序的创建，配置和维护。TwidoSoft 允许您用不同类型的语言（见 Twido 语言，p.19）创建程序，然后传递应用程序到控制器运行。

TwidoSoft

TwidoSoft是一个32位的基于Windows操作系统的软件，运行环境为个人计算机（PC）的Microsoft Windows 98第二版，Microsoft Windows 2000专业版或Microsoft Windows XP操作系统。

TwidoSoft软件的主要特点：

- 标准的Windows用户接口
- 用于Twido控制器的编程和配置
- 用于控制器的通信和控制

导言

一个可编程控制器基于控制程序来读输入，写输出，并且解决逻辑问题。创建一个 Twido 控制器的控制程序即在 Twido 的一种编程语言中写一系列的指令。

Twido 语言

下列语言可以用来创建Twido控制程序：

- 指令列表语言：
  一个指令列表程序是由布尔指令写成的一个逻辑表达序列。
- 梯形图：
  梯形图是图形方式的逻辑表达。
- Grafcet语言：
  Grafcet语言由一系列的步和转换组成。Twido支持Grafcet指令系统，但不支持图形Grafcet。

您能在个人计算机（PC）上用上述编程语言创建和编辑Twido控制程序。列表/梯形图的转换可逆特性使得您能方便地将一个程序从梯形图转换为列表或从列表转换为梯形图。

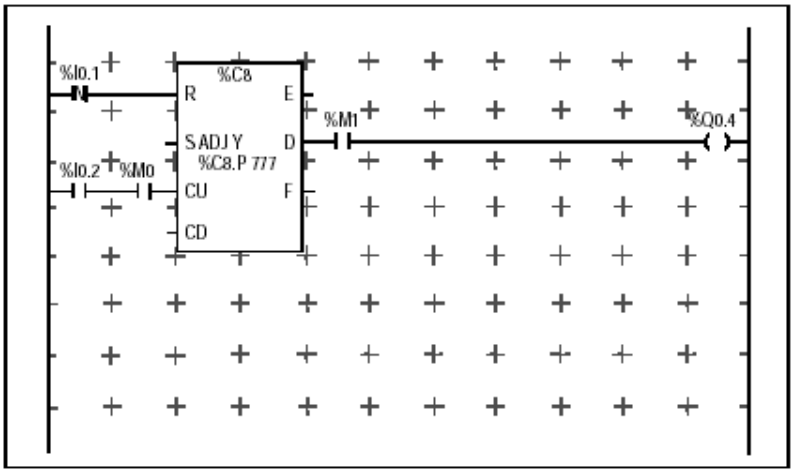
指令列表语言

指令列表语言写成的程序由一系列控制器顺序执行的指令组成。下面是一个列表程序示例：

```
0    BLK    %C8
1    LDF    %I0.1
2    R
3    LD     %I0.2
4    AND    %M0
5    CU
6    OUT_BLK
7    LD     D
8    AND    %M1
9    ST     %Q0.4
10   END_BLK
```

梯形图

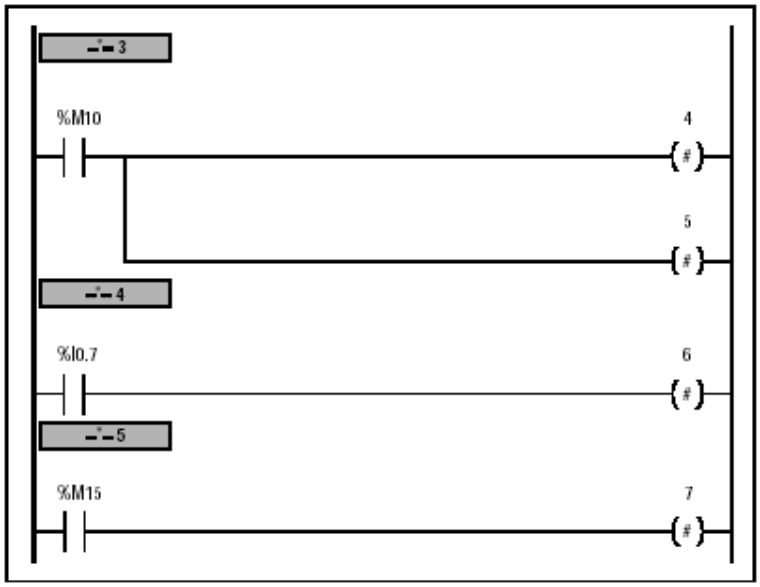
梯形图类似于表示继电器控制电路的继电器逻辑图。图形元素线圈，触点，和模块等表示指令。下面是一个梯形图示例：



Grafcet 语言

Grafcet 分析法将连续控制系统分为一系列的步，包括相关的动作，转换，和条件。下面分别是 Grafcet 指令的列表图示例和梯形图示例。

```
0     -'-     3
1     LD      %M10
2     #       4
3     #       5
4     -'-     4
5     LD      %I0.7
6     #       6
7     -'-     5
8     LD      %M15
9     #       7
10    ...
```



2

概览

本章的主题　　本章提供了用于 Twido 控制器编程的语言对象的详细资料。

本章包含了哪些内容？　　本章包含了以下主题：

语言对象生效

导言

字对象和位对象在控制器已分配它们存储空间后才有效。因此应用程序在下载到控制器之前必须用到它们。

示例

对象的有效范围是从零到此类对象的最大参数值。例如：如果存储字在您的应用程序中最大参数值是%MW9，则%MW0 到%MW9 被分配空间。该例中%MW10 无效且其内部访问和外部访问均不允许。

位对象

导言

位对象是用作操作数且为布尔指令测试的位-类型软件变量。下面是位对象列表：

- I/O位
- 内部位（存储位）
- 系统位
- 步位
- 字的抽取位

位的操作数列表

下面表格列出并描述了所有在布尔指令中用作操作数的主要位对象。

类型	描述	地址或值	最大值	写访问（1）
立即值	0 或 1（假或真）	0 或 1	—	—
输入 输出	这些位是 I/O 电气状态的“逻辑映像”。它们存储在数据存储器中且在每次程序逻辑扫描时得到更新。	%Ix.y.z (2) %Qx.y.z (2)	注解（4）	不可以 可以
内部 (存储)	内部位是程序运行时存储立即数的内存区域。 注意：未用的I/O位不能用作内部位。	%Mi	128 TWDLCAA10DRF， TWDLCAA16DRF 256 所有其它控制器	可以
系统	系统位%S0到%S127监控控制器的正确操作及应用程序的正确运行。	%Si	128	对应 i
函数模块	函数模块位对应函数模块的输出。这些输出或者直接被连接，或者用作一个对象。	%TMi.Q， %Ci.P， 等等	注解（4）	不可以（3）
可逆函数模块	用可逆编程指令BLK，OUT_BLK，和END_BLK编程的函数模块。	E，D，F，Q，TH0，TH1	注解（4）	不可以
字的抽取	一些 16 位的字中一位可被抽取为操作位。	不定	不定	不定

类型	描述	地址或值	最大值	写访问（1）
Grafcet 步	位%X1 到%Xi 对应 Grafcet 步。步位 Xi 当对应步处于活动状态时置为 1，当对应步处于非活动状态时置为 0。	%X21	62 TWDLCAA10DRF，TWDLCAA16DRF 94 TWDLCAA24DRF，模块控制器	可以

注解：

1. 被程序写或用Animation Tables Editor 写。
2. 见I/O寻址。
3. 除了位%SBRi.j 和%SCi.j能被读和写。
4. 数值由控制器模型决定。

字对象

导言

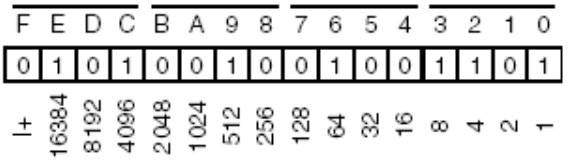
字对象是存放在数据存储区中的 16 位字，它们可表示-32768 到 32767 之间的任何整数（除了高速计数器函数模块是 0 到 65535）。

字对象举例：

- 立即数
- 内部字(%MWi) (存储字)
- 常量字(%KWi)
- I/O交换字(%IWi, %QWi)
- 系统字(%SWi)
- 功能模块(配置数据和/或运行数据)

字的格式

字的内容或值根据下述约定以 16 位二进制码（或补码）的形式存放在用户内存中：



位的位置
位的状态
位的值

在带符号的二进制码中，第 15 位根据约定用于标示值的正负：

- 第 15 位为 0：字的值为正。
- 第 15 位为 1：字的值为负（负值用二进制补码逻辑表示）。

字和立即值可以以以下形式存储和读取：

- 十进制

最小值：-32768，最大值：32767（例如，1579）

- 十六进制

最小值：16#0000，最大值：16#FFFF（例如，16#A536）

替换语法：#A536

字对象描述　下面表格是字对象的描述。

字	描述	地址或值	最大值	写访问(1)
立即值	这些整数值的格式和 16 位字一样，允许这些值赋给这些字。		—	不可以
	10 进制	-32768到32767		
	16 进制	16#0000到16#FFFF		
内部(存储)	操作期间作为“工作”字存值于数据存储区。字%MW0 和%MW255 直接被程序读或写。	%MWi	1500 *3000	可以
常量	存储常量或文字数字信息。它们的内容只能通过TwidoSoft配置来写或修改。常量字%KW0到%KW63程序中为只读。	%KWi	64 *256	可以，但是只能通过TwidoSoft
系统	这些 16 位字具有功能： • 通过读字%Swi为直接来自控制器的将到数据提供访问。 • 完成应用程序中的操作（例如，调节时间模块）。	%SWi	128	对应 i
功能模块	这些字对应功能模块的当前参数或值。	%TM2.P， %Ci.P， 等等		可以
网络交换字	赋值给远程连接控制器。这些字用于控制器间通信：			
	网络输入	%INWi.j	4 每个远程连接	不可以
	网络输出	%QNWi.j	4 每个远程连接	可以

字	描述	地址或值	最大值	写访问(1)
I/O 交换字	赋值给远程连接控制器。这些字用于控制器与 I/O 模块间通信：			
	输入	%IWi.j	注解(2)	不可以
	输出	%QWi.j	注解(2)	可以
位 抽取	可能从下列字中的 16 位中抽取一位：			
	内部	%MWi:Xk	1500*3K	可以
	系统	%SWi:Xk	128	决定于 i
	常量	%KWi:Xk	64*256	不可以
	输入	%IWi.j:Xk	注解(2)	不可以
	输出	%QWi.j:Xk	注解(2)	可以
	网络输入	%INWi.j:Xk	注解(2)	不可以
	网络输出	%QNWi.j:Xk	注解(2)	可以

注解：
1. 被程序写或用 Animation Tables Editor 写。
2. 数值由配置决定。

导言

TwidoSoft 允许您进行浮点数和双整字对象操作。

浮点数是其表达式中含有小数的数学量。(例如：3.4E+38，2.3 或 1.0)。

双整字是存放在数据存储区中的 4 字节字，包含介于-2147483648 和 +2147483647 之间的一个值。

浮点数格式及值

所用浮点格式是基于 IEEE STD 734-1985 标准（等价于 IEC 559）。其字长 32 位，对应一个小数点和浮点数值。

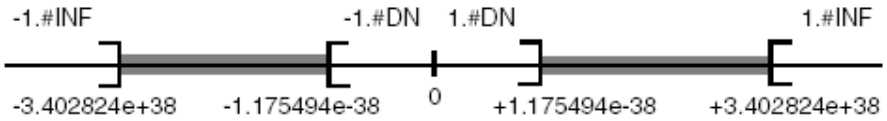
浮点值格式见下表：

位 31	位{30…23}	位{22…0}
S	指数	小数部分

上面格式的表示值由下面方程决定：

32位浮点值= $(-1)^S * 2^{(指数-127)} * 1.$小数部分

浮点值表达式中可有或没有指数，但它们一般必须有小数点（浮点）。浮点值范围从-3.402824e+38 和 -1.175494e-38 到 1.175494e-38 和 3.402824e+38（图中灰色值）。它们也包含值0，记为0.0。



当计算结果是：

- 小于-3.402824e+38，显示符号-1.#INF（表示负无穷）
- 大于+3.402824e+38，显示符号1.#INF（表示正无穷）
- 介于-1.175494e-38 和 1.175494e-38之间，近似为0.0。这两个界限之间的值不是浮点值。
- 不确定（例如负数的平方根），则显示符号1.#NAN或-1.#NAN。

该表示法精度为2-24。显示浮点数，小数点后6位阿拉伯数字即足够。

注意：

- 值“1285”将作为一个完整值翻译；为了将其作为浮点值辨识，必须记做“1285.0”

硬件兼容性

不是所有 Twido 控制器支持浮点和双字操作。

下表显示硬件兼容性：

Twido 控制器	双字支持	浮点支持
TWDLCAA40DRF	是	是
TWDLMDA40DUK	是	是
TWDLMDA40DTK	是	是
TWDLMDA20DUK	是	否
TWDLMDA20DTK	是	否
TWDLMDA20DRT	是	是
TWDLCAA24DRF	是	否
TWDLCAA16DRF	是	否
TWDLCAA10DRF	否	否

有效性检查

当结果不在有效范围之内，系统位%S18 将置为 1。

状态字%SW17 的位显示浮点操作出错的原因：

字%SW17 的不同位：

%SW17:X0	无效操作，结果不是一个数(1.#NAN 或 -1.#NAN)
%SW17:X1	非标准操作数(介于-1.175494e-38 和 1.175494e-38 之间)，结果近似为 0.0。
%SW17:X2	被 0 除，结果为无穷(-1.#INF 或 1.#INF)
%SW17:X3	结果绝对值大于+3.402824e+38，视为无穷大(-1.#INF 或 1.#INF)
%SW17:X4	保留
%SW17:X5	保留

该字在系统冷启动或程序重复使用时置为0。

浮点和双字描　下表是浮点和双字的描述：
述

对象类型	描述	地址	最大值	写访问	索引表
立即值	32 位同样格式的整数或小数对象。	—	[—]	不可以	—
内部浮点	对象用于操作过程中存储值于数据存储区中。	%MFi	1499*2998	可以	%MFi[索引]
内部双字		%MDi	1499*2998	可以	%MDi[索引]
浮点常量	用于存储常量	%KFi	63*254	可以，但只能通过 TwidoSoft	%KFi[索引]
双字常量		%KDi	63*254	可以，但只能通过 TwidoSoft	%KDi[索引]

对象之间重迭的可能性

单字长，双字长和浮点字均存储于同一存储区域的数据空间。这样，浮点字%Mfi 和双字%Mdi 与单字%Mwi 和%MWi+1 相对应(字%Mwi 包含最少有意义位且字%MWi+1 包含字%Mfi 的最多有意义位)。

下表显示了浮点和内部双字是怎样重迭的：

浮点双字	奇数地址	内部字
%MF0 / %MD0		%MW0
%MF2 / %MD2	%MF1 / %MD1	%MW1
%MF4 / %MD4	%MF3 / %MD3	%MW2
...	...	%MW3
%MFi+1 / %MDi+1	%MFi / %MDi	%MW4
		%MW5
		...
		%MWi
		%MWi+1

下表显示了浮点和双字常量是怎样重迭的：

浮点双字	奇数地址	内部字
%KF0 / %KD0		%KW0
%KF2 / %KD2	%KF1 / %KD1	%KW1
%KF4 / %KD4	%KF3 / %KD3	%KW2
...	...	%KW3
%KFi+1 / %KDi+1	%KFi / %KDi	%KW4
		%KW5
		...
		%KWi
		%KWi+1

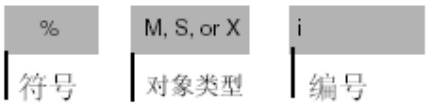
示例：

%MF0对应 %MW0 和 %MW1。%KF543*43对应 %KW543*43 和 %KW544*44。

位对象寻址

语法

用下面格式寻址内部位，系统位，和步位对象：



描述

下面表格是对寻址格式中各元素的描述。

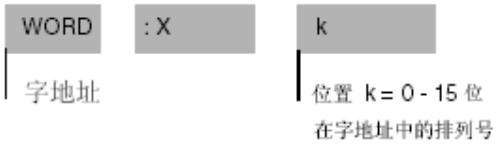
组	项	描述
符号	%	百分比符号一般用于软件变量前。
对象类型	M	内部位在程序运行时存储中间值。
	S	系统位提供控制器状态和控制信息。
	X	步位提供步的活动状态。
编号	i	最大编号值取决于配置对象的编号。

位对象寻址示例：

- %M25 = 内部位编号 25
- %S20 = 系统位编号 20
- %X6 = 步位编号 6

来自字的抽取位

通过 TwidoSoft 可以从字的 16 位中抽取一位。根据下面语法字的地址加上了抽取位的排列号。



示例：

- %MW5:X6 = 内部字%MW5 的第 6 位
- %QW5.1:X10 = 输出字%QW5.1 的第 10 位

字对象寻址

导言

除了输入/输出寻址(见输入/输出寻址，p.40)和功能模块(见功能模块对象，p.43)，字对象寻址遵循下面格式。

语法

下面格式用于寻址内部字，常量字，和系统字：

描述

下面表格是对寻址格式中各元素的描述。

组	项	描述
符号	%	百分比符号一般用于内部地址前。
对象类型	M	内部字在程序运行时存储中间值。
	K	常量字存储常量值或文字数字信息。它们的内容只能通过 Twidosoft 写或修改。
	S	系统字提供控制器状态和控制信息。
格式	W	16 位字。
编号	i	最大编号值取决于配置对象的编号。

字对象寻址示例：

- %MW15 = 内部字编号 15
- %KW26 = 常量字编号 26
- %SW30 = 系统字编号 30

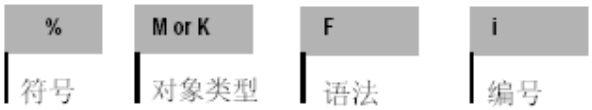
浮点对象寻址

导言

除了输入/输出寻址(见输入/输出寻址，p.40)和功能模块(见功能模块对象，p.43)，浮点对象寻址遵循下面格式。

语法

下面格式用于寻址内部和常量浮点对象：



描述

下面表格是对寻址格式中各元素的描述。

组	项	描述
符号	%	百分比符号一般用于内部地址前。
对象类型	M	内部浮点对象在程序运行时存储中间值。
	K	浮点常量存储常量值。它们的内容只能通过 Twidosoft 写或修改。
语法	F	32 位对象。
编号	i	最大编号值取决于配置对象的编号。

浮点对象寻址示例：

- %MF15 = 内部浮点对象编号 15
- %KF26 = 常量浮点对象编号 26

双字对象寻址

导言

除了输入/输出寻址(见输入/输出寻址，p.40)和功能模块(见功能模块对象，p.43)，双字对象寻址遵循下面格式。

语法

下面格式用于寻址内部和常量双字：

描述

下面表格是对寻址格式中各元素的描述。

组	项	描述
符号	%	百分比符号一般用于内部地址前。
对象类型	M	内部双字在程序运行时存储中间值。
	K	常量双字存储常量值或文字数字信息。它们的内容只能通过 Twidosoft 写或修改。
语法	D	32 位双字。
编号	i	最大编号值取决于配置对象的编号。

双字对象寻址示例：

- %MD15 = 内部双字编号 15
- %KD26 = 常量双字编号 26

输入/输出寻址

导言

Twido 配置中每个输入/输出(I/O)点有一个唯一地址：例如，地址“%I0.0.4”表示*基本控制器的输入 4。

I/O地址可赋值给下列硬件：

- 设定为远程连接主机的控制器
- 设定为远程I/O的控制器
- 扩展I/O模块

TWDNOI10M3 AS-I总线接口模块对其从设备的I/Os有一个特殊的地址系统(见AS-i V2总线所连从设备I/Os寻址，p.180)。

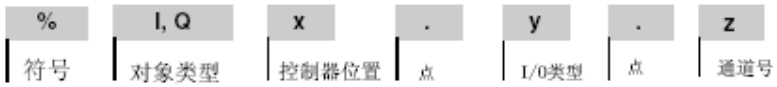
输出或线圈的多重涉及

一个程序中，您可以对一个输出或线圈有多重涉及。但只有最后的结果才能更新硬件的输出。例如，%Q0.0.0 可在一个程序中用到多次，并且这种多次出现不会得到任何警告。因此有必要确定只有一个因素给定输出的所需状态。

	警告
	意外操作 不提供任何重复输出检查或警告。您在应用程序中改变输出或线圈之前，请仔细阅读它们的使用。 如果不注意这个警告，会导致伤害或设备损坏。

格式

用下面格式寻址输入/输出。



用下面格式寻址输入/输出交换字。

描述

下面表格描述了 I/O 寻址格式。

组	项	值	描述
Symbol	%	—	百分比符号一般用于内部地址前。
Object type	I	—	输入。控制器或扩展I/O模块输入电气状态的“逻辑映像”。
	Q	—	输出。控制器或扩展I/O模块输出电气状态的“逻辑映像”。
Controller position	x	0 1—7	主控制器（远程连接主机）。 远程控制器（远程连接从机）。
I/O Type	y	0 1—7	基本 I/O（控制器本地 I/O）。 扩展 I/O 模块。
Channel Number	z	0—31	控制器或扩展 I/O 模块的 I/O 通道号。可用 I/O 点的编号取决于控制器模型和扩展 I/O 模块类型。

示例

下表是 I/O 寻址的一些示例。

I/O 对象	描述
%I0.0.5	基本控制器 5 号输入点（本地 I/O）。
%Q0.3.4	3号本地控制器扩展I/O模块4号输出点（扩展I/O）。
%I0.0.3	基本控制器 3 号输入点。
%I3.0.1	3 号远程连接 I/O 控制器 1 号输入点。
%I0.3.2	3 号本地控制器扩展 I/O 模块 2 号输入点。

网络寻址

导言

Twido远程连接网络中的应用程序数据通过网络字%INW和%QNW在对等控制器和主控制器间得到交换。详细资料请见通信，p.87。

格式

下面格式用于网络寻址。

格式描述

下面表格是网络寻址格式描述。

组	项	值	描述
符号	%	—	百分比符号一般用于内部地址前。
对象类型	IN	—	网络输入字。数据传输从主机到从机。
	QN	—	网络输出字。数据传输从从机到主机。
格式	W	—	一个 16 位字。
控制器位置	x	0 1—7	主控制器（远程连接主机）。 远程控制器（远程连接从机）。
字	j	0—3	每个对等控制器用一到四个字与主控制器进行数据交换。

示例

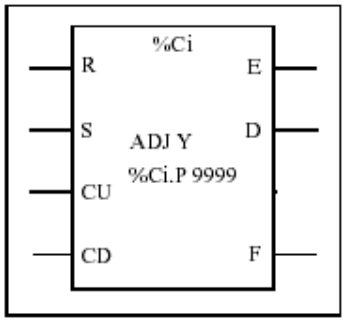
下表是网络寻址的一些示例。

网络对象	描述
%INW3.1	3 号远程控制器 1 号网络字。
%QNW0.3	基本控制器3号网络字。

功能模块对象

导言　　功能模块提供了可供程序访问的位对象和特殊字。

功能模块示例　　下面是一个计数器功能模块。



加/减计数器模块

位对象　　位对象对应模块输出。布尔测试指令能用下面任一方法访问这些位：

- 直接方式(例如，LD E)，如果它们在可逆编程中与模块有线连接（见标准功能模块编程原则，p.280）。
- 通过指定模块类型（例如，LD %Ci.E）。

输入由指令表访问。

字对象　　字对象对应指定的参数和值如下：

- 模块配置参数：一些程序可访问参数（例如，预选参数），和一些程序非访问参数（例如，时基）。
- 当前值：例如，%Ci.V，当前计数值。

程序可访问对象　　请参见下面相应部分列出的程序可访问对象。

- 针对基本功能模块，见基本功能模块，p.278。
- 针对高级功能模块，见与高级功能模块相关的位和字对象，p.334。

导言

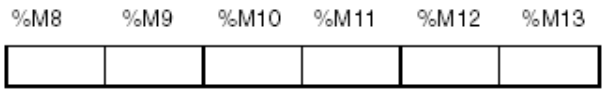
结构化对象是邻近对象的联合。Twido支持下列结构化对象：

- 位串
- 字表
- 双字表
- 浮点字表

位串

位串是指一系列类型相同的相邻对象位，并被定义为长度（L）。

例如：位串%M8:6



注意：%M8:6 是可接受的（8 是 8 的倍数），但%M10:16 是不可接受的（10 不是 8 的倍数）。

位串可被用于赋值指令（见赋值指令，p.305）。

位的可用类型

位串中位的可用类型：

类型	地址	最大值	写访问
离散输入位	%I0.0:L或%I1.0:L (1)	0<L<17	不可以
离散输出位	%Q0.0:L或%Q1.0:L (1)	0<L<17	可以
系统位	%Si:L，i为8的倍数	0<L<17和 i+L<=128	取决于i
Grafcet步位	%Xi:L，i为8的倍数	0<L<17和 i+L<=95 (2)	可以(通过程序)
内部位	%Mi:L，i为8的倍数	0<L<17和 i+L<=256 (3)	可以

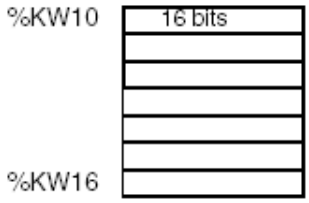
注解：

1. 位串中I/O位只有0到16可读。对于24输入的控制器和32 I/O模块，位串中高于16的位将不可读。
2. TWWDLCAA10DRF 和 TWDLCAA16DRF的i+L最大值是62
3. TWWDLCAA10DRF 和 TWDLCAA16DRF的i+L最大值是128

字表

字表是指一系列类型相同且相邻的字，并被定义为长度（L）。

例如：字表%KW10:7



字表可被用于赋值指令（见赋值指令，p.305）。

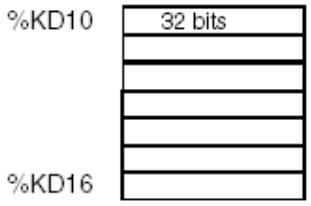
字的可用类型

字表中字的可用类型：

类型	地址	最大值	写访问
内部字	%MWi:L	0<L<256 且 i+L<=1500	可以
常量字	%KWi:L	0<L和i+L<=64	不可以
系统字	%SWi:L	0<L和i+L<=128	取决于i

双字表

双字表是指一系列类型相同且相邻的双字，并被定义为长度（L）。
例如：双字表%KD10:7



双字表可被用于赋值指令（见赋值指令，p.305）。

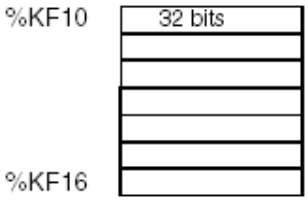
双字的可用类型

双字表中双字的可用类型：

类型	地址	最大值	写访问
内部双字	%MDi:L	0<L<256 且 i+L<=1499	可以
常量双字	%KDi:L	0<L和i+L<=63	不可以

浮点字表

浮点字表是指一系列类型相同且相邻的浮点字，并被定义为长度（L）。
例如：浮点字表%KF10:7



浮点字表可被用于赋值指令（见高级指令）。

浮点字的可用类型

浮点字表中浮点字的可用类型：

类型	地址	最大值	写访问
内部浮点字	%MFi:L	0<L<256 且 i+L<=1499	可以
常量浮点字	%KFi:L	0<L和i+L<=63	不可以

索引对象

导言

索引字指的是含有索引对象地址的单字，双字或浮点。对象寻址方式有两种：

- 直接寻址
- 索引寻址

直接寻址

当程序写完之后，对象的直接地址就被设定和定义。

例如：%M26 此内部位的直接地址是 26。

索引寻址

对象的索引地址通过给对象的直接地址添加一个索引，提供了一个修改对象地址的方法。索引的内容被加到对象的直接地址中去。索引由内部字%MWi 定义。“索引字”的数量没有限制。

例如：%MW108[%MW2]字的地址由直接地址 108 加上字%MW2 的内容组成。

如果字%MW2 的值是 12，则写入%MW108[%MW2]等价于写入%MW120（180 加 12）。

可索引寻址的对象

下面是可以索引寻址的对象类型。

类型	地址	最大值	写访问
内部字	%MWi[MWj]	0<= i+%MWj<1500	可以
常量字	%KWi[%MWj]	0<= i+%MWj<64	不可以
内部双字	%MDi[MWj]	0<= i+%MWj<1499	可以
常量双字	%KDi[%MWj]	0<= i+%MWj<63	不可以
内部浮点	%MFi[MWj]	0<= i+%MWj<1499	可以
常量浮点	%KFi[%MWj]	0<= i+%MWj<63	不可以

索引对象可被用于赋值指令（见赋值指令，p.305 单字和双字）和比较指令（见比较指令，p.311 单字和双字）。这种寻址使得通过修改程序中索引对象的内容，可以连续扫描一系列相同类型的对象（如内部字和常量）。

索引溢出系统位 **%S20**

当索引对象的地址超出此类对象存储区域的限制，就会发生索引溢出。概括如下：

- 对象地址加索引内容小于 0。
- 对象地址加索引内容大于程序直接引用字的最大值。最大值是 1499（对字%MWi）或 63（对字%KWi）。索引溢出事件发生后，系统将系统位%S20 置为 1，且该对象索引值赋为 0。

注意：用户有责任对任何溢出进行监测。用户程序必须读位%S20 并作可能的处理。用户必须确认将其复位到 0。

%S20（初始状态=0）：

- 索引溢出发生：系统将其置为 1。
- 溢出确认：用户在修改索引后，将其置为 0。

符号化对象

导言

您能用名字或用户化的记忆符号来标注Twido软件语言对象。符号的使用使得程序逻辑可以快速检查和分析，并且极大地简化了应用程序的开发和测试。

示例

例如，WASH_END 这个符号可用来表示一个定时器功能模块中一个冲洗周期的结束。记忆这个名字的意义将比记忆一个程序地址如%TM3的意义容易得多。

符号定义原则

符号定义原则如下：

- 最多 32 个字符。
- 只能使用字母(A-Z)，数字(0 -9)，和下划线(_)。
- 第一个字符必须是字母或重音字符。您不能使用百分号(%)。
- 不能使用空格和特殊字符。
- 不区分大小写。例如，Pump1 和 PUMP1 视为相同符号，在一个程序中只允许用一次。

符号编辑

符号在符号编辑器中被定义并与语言对象相关联。程序里这些符号和其注释存储在 PC 硬盘而不是控制器里。因此，它们不随应用程序传递给控制器。

用户存储器

3

概览

本章的主题　　本章描述了 Twido 用户存储器的结构和用法。

本章包含了哪些内容？　　本章包含了以下主题：

用户存储器结构

导言

应用程序可以访问的控制器存储器分为两个不同的集：

- 位值
- 字值（16位符号数）

位存储器

位存储器位于处理器的内置 RAM。它含有 128*个对象单元。

存储字

字存储器（16 位）支持：

- 动态字：运行存储（仅存储于 RAM）。
- 存储字(%MW)：动态系统数据和系统数据。
- 程序：任务描述符和执行码。
- 配置数据：常量字，初始值，和输入/输出配置。

存储器存储类型

下面是 Twido 控制器存储器存储的不同格式。

- 随机存取存储器

内部暂时存储器：包含动态字，存储字，程序和动态数据。

- EEPROM

一个完整的32KB EEPROM提供内部程序和数据备份。保护程序不因电池失效或能量储运消耗超过30天而导致崩溃。包含程序和配置数据。保存最多可达512个存储字。如果使用64K扩展存储卡程序将不再备份在这里，且Twido已配置成接受64K扩展存储卡

- 32K备份卡

一个可选择的外部卡，用来保存程序及传递程序给其它Twido控制器。能用来更新控制器RAM中的程序。包含程序和常量，但不包含存储字。

- 64K扩展存储卡

一个可选择的外部卡，可存储多达64K的程序。卡必须插在控制器中卡中程序才可使用。

存储器挽救

控制器的程序和存储字可以保存在：

- RAM（*完全充电的电池最多可达 30 天）
- EEPROM（最大容量 32 KB）

RAM中程序丢失（或没有电池）时，程序能自动从EEPROM存储器传递给RAM存储器。通过TwidoSoft也可手动完成传递。

存储器配置　下表描述了 Twido 控制器存储器可能的配置类型。

存储器类型	一体型控制器			模块型控制器		
	10DRF	**16DRF**	**24DRF**	**20DUK 20DTK**	**20DRT**	**40DUK 40DTK**
内部RAM	10KB	32KB	32KB	32KB	32KB	32KB
支持扩展存储器*					64KB	64KB
最大程序容量	8KB	16KB	32KB	32KB	32KB 或 64KB*	32KB 或 64KB*
最大外部备份	8KB	16KB	32KB	32KB	32KB	32KB

注意：*TWDLMDA20DRT，TWDLMDA40DUK，和TWDLMDA40DTK控制器可安装64KB扩展存储卡使程序存储器扩展至64KB。程序运行时确保卡已安装好。

没有备份卡和扩展存储器时的备份和恢复

导言

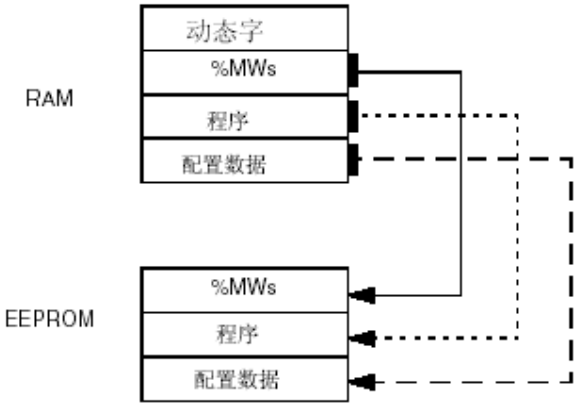
下面信息详细叙述了模块型和一体型控制器在没有备份卡和扩展存储器插入情况下的备份和恢复。

概览

Twido 程序，存储字和配置数据可用控制器内部 EEPROM 备份。因为保存程序到内部 EEPROM 将清除以前所有备份的存储字，因此程序和已配置的存储字必须备份。动态数据可存储在存储字里然后备份到 EEPROM。存储字只能在程序之后保存到内部 EEPROM。

存储器结构

这里是控制器的存储器结构图。箭头显示了哪些内容可从 RAM 备份到 EEPROM：



程序备份

这里是程序备份到 EEPROM 的步骤：

步骤	动作
1	下面必须为真： RAM 中有一个有效程序。
2	从Twid软件窗口打开菜单“Controller”，找到“Backup”并点击它。

程序恢复

上电时有一种情况程序将从EEPROM恢复到RAM（假设没有卡和扩展存储器存在）：

- RAM中的程序无效。

从EEPROM手动恢复程序：

- 从Twid软件窗口打开菜单“Controller”，找到“Restore”并点击它。

数据**(%MWs)**备份

这里是备份数据（存储字）到 EEPROM 的步骤：

步骤	动作
1	下面必须为真： RAM 中有一个有效程序(%SW96:X6=1)。 相同的有效程序已备份到 EEPROM。 程序已配置存储字。
2	将%SW97置为将要保存的存储字的长度。 注意：长度不能超过存储字的配置长度，且必须大于0，不超过512。
3	将%SW96:X0置为1。

数据**(%MWs)**恢复

手动置系统位%S95 为 1 即恢复%MWs。

但必须确保下面为真：

- EEPROM 存在有效备份程序
- RAM 中程序与 EEPROM 备份程序匹配
- 备份的存储字有效

使用 32K 备份卡备份和恢复

导言

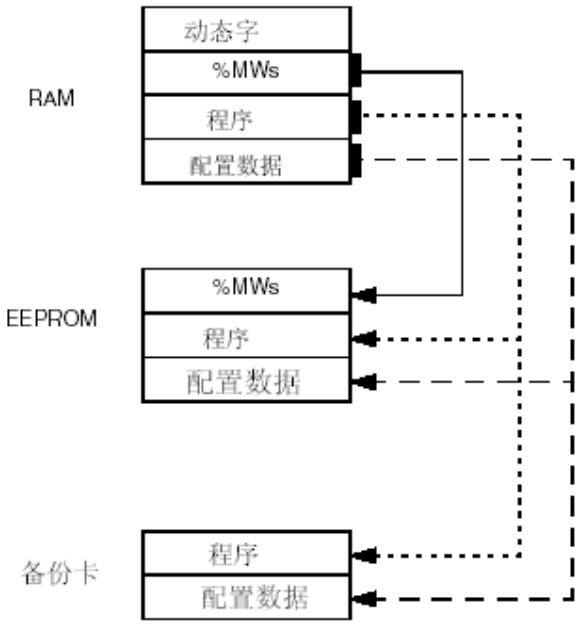
下面信息详细叙述了模块型和一体型控制器怎样使用32K备份卡进行备份和恢复。

概览

备份卡用来保存程序及传递程序到其它 Twido 控制器。一旦程序安装或保存完毕，卡应从控制器卸载并放到旁边。卡只能保存程序和配置字（%MWs 不能保存在 32K 备份卡里）。动态数据可存储在存储字里然后备份到 EEPROM。当程序安装完成，任何在安装之前备份到 EEPROM 的%MWs 都将丢失。

存储器结构

这里是控制器附接备份卡的存储器结构图。箭头显示了哪些内容可从 RAM 备份到 EEPROM 和卡：



程序备份

这里是程序备份到备份卡的步骤：

步骤	动作
1	控制器断电。
2	插入备份卡。
3	控制器上电。
4	从Twid软件窗口打开菜单“Controller”，找到“Backup”并点击它。
5	控制器断电。
6	从控制器卸载备份卡。

程序恢复

下面是加载备份卡保存的程序到控制器：

步骤	动作
1	控制器断电。
2	插入备份卡。
3	控制器上电。 （如果配置了自动运行，您必须重启进入运行模式。）
4	控制器断电。
5	从控制器卸载备份卡。

数据**(%MWs)**备份

这里是备份数据（存储字）到 EEPROM 的步骤：

步骤	动作
1	下面必须为真： RAM 中有一个有效程序(%SW96:X6=1)。 相同的有效程序已备份到 EEPROM。 程序已配置存储字。
2	将%SW97置为将要保存的存储字的长度。 注意：长度不能超过存储字的配置长度，且必须大于0，不超过512。
3	将%SW96:X0置为1。

数据**(%MWs)**恢复

手动置系统位%S95 为 1 即恢复%MWs。

但必须确保下面为真：

- EEPROM 存在有效备份程序
- RAM 中程序与 EEPROM 备份程序匹配
- 备份的存储字有效

64K 扩展存储卡的使用

导言

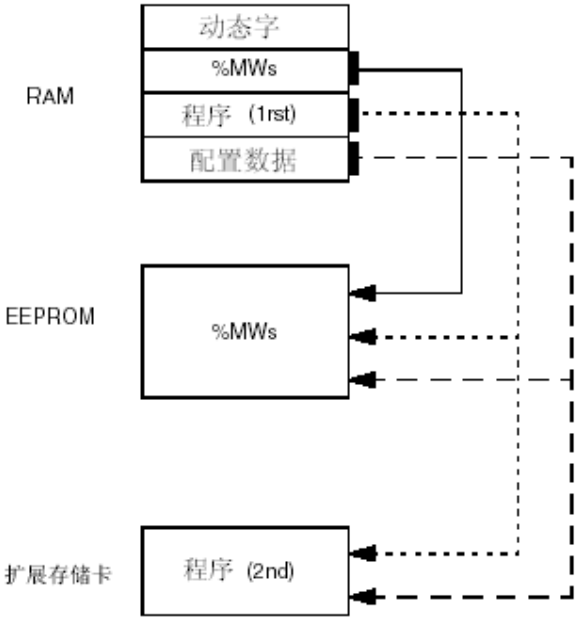
下面信息详细叙述了模块型控制器安装64K扩展存储卡后存储器的功能使用。

概览

64K 扩展存储卡将 Twido 控制器的程序存储容量从 32K 扩展到 64K。扩展程序使用时卡必须插在控制器里。如果卡被卸载，控制器将进入停止状态。存储字仍然备份到控制器的 EEPROM。动态数据可存储在存储字里然后备份到 EEPROM。64K 扩展存储卡的上电动作和 32K 备份卡相同。

存储器结构

这里是控制器附接扩展内存卡的存储器结构图。箭头显示了哪些内容可从 RAM 备份到 EEPROM 和 64K 扩展内存卡：



软件配置和扩展存储器安装

在写入您的扩展程序之前，您必须安装到 64K 扩展内存卡到控制器上。下面是操作步骤：

步骤	动作
1	在 Twido 软件窗口硬件选择菜单下进入“TWDXCPMFK64”。
2	控制器断电。
3	插入64K扩展内存卡。
4	控制器上电。

保存您的程序。

一旦您的 64K 扩展内存卡安装完毕且您的程序已写完：

- 从*Twidosoft 软件窗口打开菜单“Controller”，找到“Backup”并点击它。

数据**(%MWs)**备份

这里是备份数据（存储字）到 EEPROM 的步骤：

步骤	动作
1	下面必须为真： 一个有效程序存在 程序已配置存储字。
2	将%SW97置为将要保存的存储字的长度。 注意：长度不能超过存储字的配置长度，且必须大于0，不超过512。
3	将%SW96:X0置为1。

数据**(%MWs)**恢复

手动置系统位%S95 为 1 即恢复%MWs。

但必须确保下面为真：

- EEPROM 存在有效备份程序
- 备份的存储字有效

4

概览

本章的主题

本章描述了控制器工作模式和程序执行的循环和周期。包含了有关能量损耗和恢复的详细信息。

本章包含了哪些内容?

本章包含了以下主题:

循环扫描

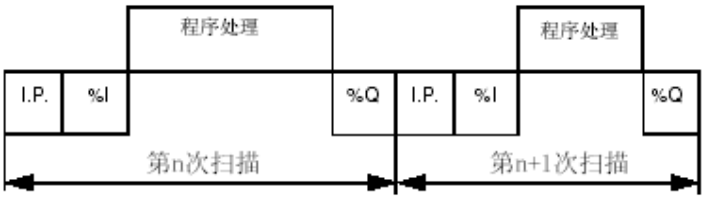
导言

扫描循环包括连接一个接一个的控制循环到一起。在执行输出更新（任务循环的第三阶段）之后，系统执行自己任务的一个特定编号且立即触发另一个任务循环。

注意：用户程序的扫描时间由控制器的看门狗定时器监测且不能超过 500 ms。否则将出错并导致控制器在暂停模式下立即停止。该模式将强制输出为它们默认的后退状态。

运行

下图显示了循环扫描时间的运行状态。



循环状态描述

下表描述了循环状态。

地址	状态	描述
I.P.	内部处理	系统后台监控控制器（管理系统位和字，更新当前定时器值，更新状态指示灯，检测 RUN/STOP 开关，等等）及处理 TwidoSoft 请求（修改和激活）。
%I, %IW	输入采集	将离散状态和程序特殊模块输入写入存储器。
-	程序处理	运行用户应用程序。
%Q, %QW	输出更新	写与离散和程序特殊模块关联的输出位或字。

工作模式

控制器在运行状态，处理器执行：

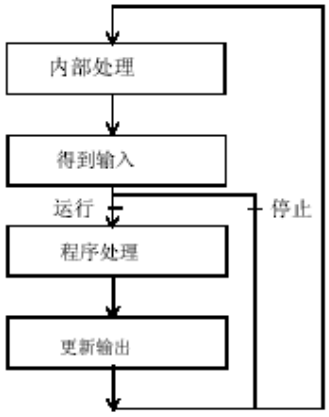
- 内部处理
- 输入采集
- 应用程序运行
- 输出更新

控制器在停止状态，处理器执行：

- 内部处理
- 输入采集

图例

下面图例显示了循环的运行过程。



循环的检查

看门狗检查循环。

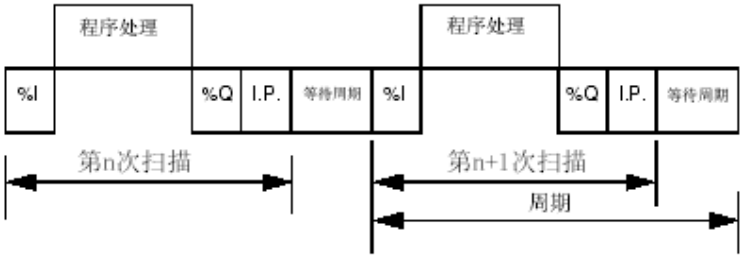
周期扫描

导言

在此运行方式下，输入采集，应用程序运行，及输出更新根据配置定义时间（2-150 ms）被周期性地执行。
在控制器扫描开始时，一个定时器开始倒计时，其值在配置定义周期里初始化。控制器扫描必须在定时结束之前完成扫描并开始一个新的扫描。

运行

下图显示了周期扫描时间的运行状态。



运行状态描述

下表描述了运行状态。

地址	状态	描述
I.P.	内部处理	系统后台监控控制器（管理系统位和字，更新当前定时器值，更新状态指示灯，检测 RUN/STOP 开关，等等）及处理 TwidoSoft 请求（修改和激活）。
%I, %IW	输入采集	将离散状态和程序特殊模块输入写入存储器。
-	程序处理	运行用户应用程序。
%Q, %QW	输出更新	写与离散和程序特殊模块关联的输出位或字。

工作模式

控制器在运行状态，处理器执行：

- 内部处理
- 输入采集
- 应用程序运行
- 输出更新

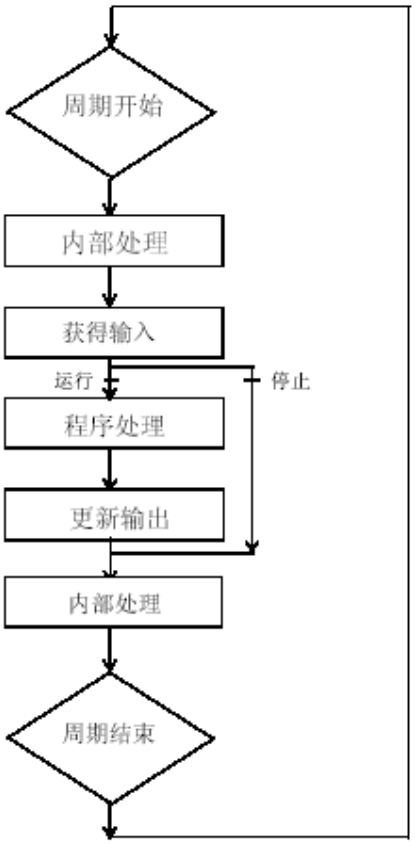
如果周期没有结束，处理器将完成操作循环直至内部处理周期结束。如果运行时间比分配的周期长，则控制器将系统位%S19 置为 1 表示周期被超出。继续处理且运行完毕。但是不能超过看门狗的时间限制。后续扫描在写输出扫描后以后台运行的方式被连接进来。

控制器在停止状态，处理器执行：

- 内部处理
- 输入采集

图例

下面图例显示了操作循环。



循环的检查

可执行两种检查：

- 周期溢出
- 看门狗

扫描时间检查

概要

任务循环由看门狗定时器 Tmax（任务循环的最大持续时间）所监测。它允许显示程序错误（无限循环，等等）且保证输出刷新的最大持续时间。

软件看门狗（周期或循环运行）

周期或循环运行中，触发看门狗将导致一个软件错误。应用程序进入暂停状态且系统位%S11 被置为 1。有必要重启任务与 Twido 软件连接以便分析错误原因，修改程序改正错误，然后重启程序运行。

注意：暂停状态是指由于程序软件错误如扫描溢出导致程序立即停止。数据保持当前值，以便错误原因分析。程序在处理指令处停止。与控制器的通信被打开。

周期运行检查

周期运行中附加检查用来检测周期被超出：

- %S19指示周期被超出。设为：
  - 1 当扫描时间高于任务周期时由系统设定，
  - 0 由用户设定。
- %SW0包含周期值(0-150 ms)。它：
  - 冷启动时由配置选择值初始化，
  - 能被用户修改。

主任务运行时间使用

下面系统字用于控制器扫描循环时间信息：

- **%SW11** 最大看门狗时间初始化（10 到 500 ms）。
- **%SW30** 包含最近一个控制器扫描循环的执行时间。
- **%SW31** 包含自最近一次冷启动以来最长的一个控制器扫描循环的执行时间。
- **%SW32** 包含自最近一次冷启动以来最短的一个控制器扫描循环的执行时间。

注意：这些不同信息也可从配置编辑器查到。

导言

Twido 软件用于考虑三类主要工作模式组：

- 检查
- 运行或生产
- 停止

从 **Grafcet** 开始

这些不同的工作模式可从下面 Grafcet 方法的开始或使用得到：

- Grafcet 初始化
- 步的预置
- 情形保持
- 图表固定

预处理和系统位的使用保证了有效的工作模式管理，而不使用户程序复杂和过载。

Grafcet 系统位

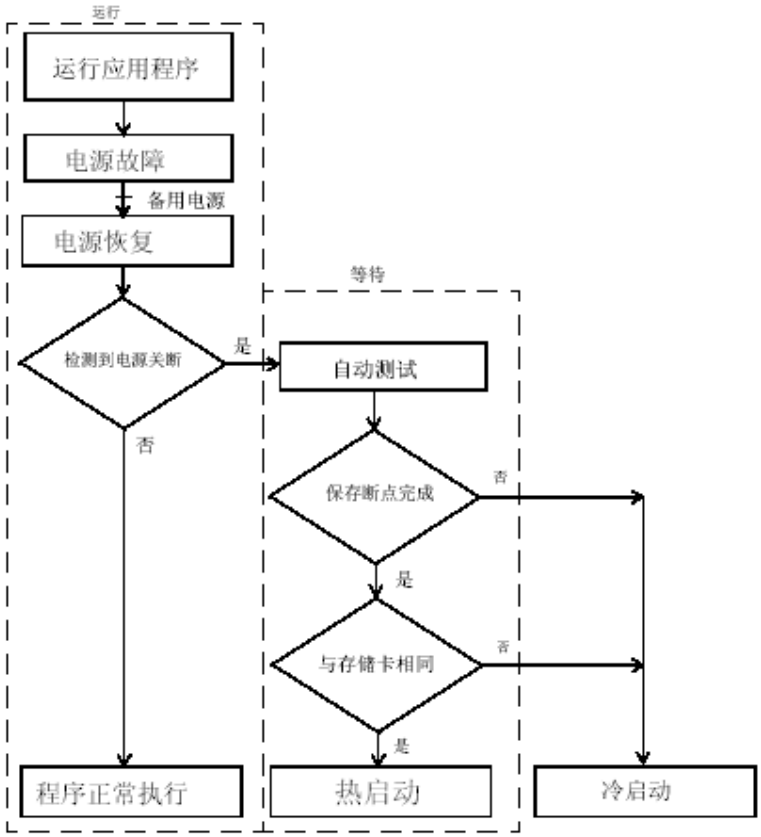
位%S21，%S22 和 %S23仅为预处理保留。这些位被系统自动复位。它们必须也只能被置位指令**S**写。下表提供了与Grafcet相关的系统位：

位	功能	描述
%S21	GRAFCET 初始化	一般置为 0，下面将其置为 1： ❙ 冷启动，**%S0=1**； ❙ 用户，仅在部分预处理程序中，用置位指令 **S %S21** 或设置线圈**-(S)-%S21**。 结果： ❙ 释放所有活动步。 ❙ 激活所有初始步。
%S22	GRAFCET 复位	一般置为 0，能且只能被程序预处理时置为 1。 结果： ❙ 释放所有活动步。 ❙ 扫描已停止的时序处理。
%S23	GRAFCET 预置和固定	一般置为 0，能且只能被程序预处理时置为 1。 ❙ 通过设置%S22为1预置。 ❙ 通过一系列S Xi指令预置将要活动的步。 ❙ 通过设置%S23为1能够预置。 固定情形： ❙ 在初始情形：通过程序将%S21维持为1。 ❙ 在一个“空”情形：通过程序将%S22维持为1。 ❙ 在一个情形决定于将%S23维持为1。

电源中断和电源恢复处理

图例

下面图例显示了系统检测到的各种电源重启。如果中断时间小于电源供应转换时间（交流电约为 10ms，直流电 1ms），程序将忽略并正常运行。



注意：断点将保存在电池备份RAM中。上电时，系统检查电池和存储的断点状态以决定是否可视为热启动。

运行/停止输入位与自动运行

运行/停止输入位的优先级比“Automatic Start in Run”选项高，该选项可从扫描模式对话框得到。如果运行/停止位被设置，控制器在电源恢复时将在运行模式下重启。

控制器模式取决如下：

运行/停止输入位	Automatic Start in Run	结果状态
0	0	停止
0	1	停止
上升沿	忽略	运行
1	忽略	运行
软件没有配置	0	停止
软件没有配置	1	运行

注意：对所有软件版本V1.0的一体型控制器，如果控制器在电源中断时处于运行模式，且“Automatic Start in Run”标志未被扫描模式对话框设置，控制器在电源恢复时将以停止模式重启。否则将执行冷重启。

注意：对所有软件版本V1.0的模块型和一体型控制器，如果控制器的电池在电源中断时工作正常，控制器将以电源中断时的工作模式启动。电源恢复时，扫描模式对话框设置的“Automatic Start in Run”标志对控制器模式不起影响。

操作

下表描述了电源中断的处理阶段。

阶段	描述
1	在电源中断事件中系统存储应用程序断点和中断时间。
2	所有输出被置为可靠状态（0）。
3	电源恢复时，被保存的断点用来与一个进程比较，该进程定义了开始到运行的类型： ❙ 如果应用程序断点改变（系统断点丢失或新的应用程序），控制器初始化应用程序：冷重启（一体型体系）。 ❙ 如果应用程序断点相同，控制器重启且不初始化数据：热重启。

热重启处理

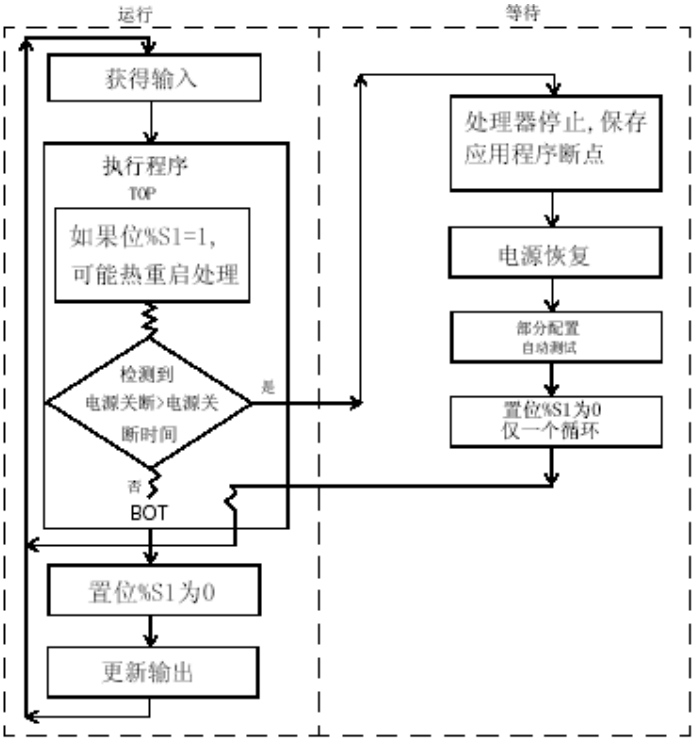
热重启原因

热重启将发生：

- 当电源恢复且应用程序断点没有丢失，
- 当位**%S1** 被程序置为状态 1，
- 当控制器处于停止模式时操作显示

图例

下图描述了运行模式下的热重启操作。



程序执行的重启

下表描述了热重启后运行程序的重启阶段。

阶段	描述
1	程序执行从电源中断前的相同环境继续开始，但不更新输出。 注意：只有来自用户代码的相同环境可被重启。系统代码（例如，输出更新）不能被重启。
2	重启循环结束时，系统： ❙ 释放被保留的应用程序（并且在调试情况下引起应用程序停止） ❙ 讯息重新初始化
3	系统执行重启循环时： ❙ 置位**%S1**（热启动指示位）和**%S13**（运行中的第一次循环）为 1 重启任务 ❙ 第一个任务循环结束时重置位**%S1** 和**%S13** 到 0

热启动过程

热启动事件中，如果需要一个特殊的应用处理，任务循环的开始必须检测位**%S1**，并且调用相应的程序。

电源故障后的输出

一旦检测到电源损耗，输出将被置为（默认）可靠状态（0）。
当电源恢复，输出将处于上次的状态直到它们被任务再次更新。

冷启动处理

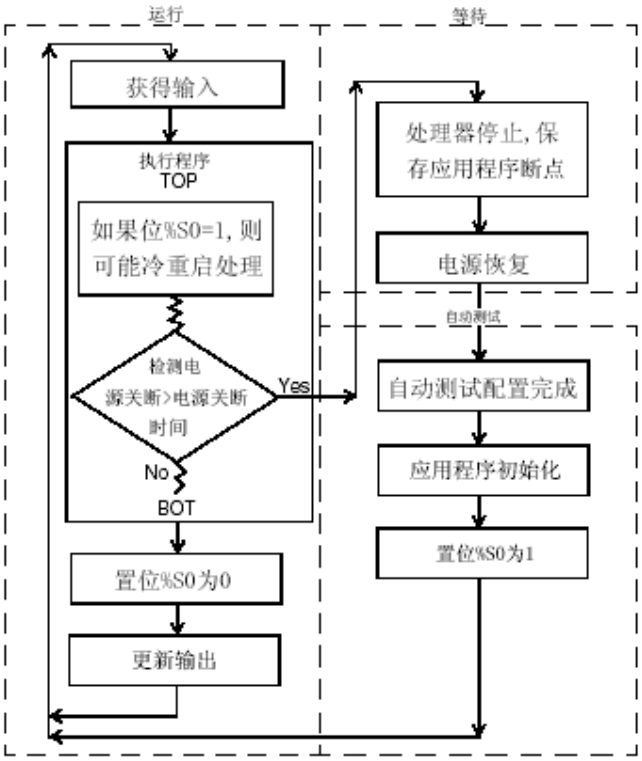
冷启动原因

冷启动将发生：

- 当装载新的应用程序到 RAM
- 当电源恢复且应用程序断点丢失
- 当系统位**%S0** 被程序置为状态 1，
- 当控制器处于停止模式时操作显示

图例

下图描述了运行模式下的冷启动操作。



操作

下表描述了冷启动后运行程序的重启阶段。

阶段	描述
1	启动时控制器处于运行模式。 若冷重启出错而停止，系统将强制冷重启。 程序执行在任务的开始重新启动。
2	系统： • 重置内部位和字及 I/O 映像到 0 • 初始化系统位和字 • 从配置数据初始化功能模块
3	对第一个重启循环，系统： • 置位**%S0**（冷启动指示位）和**%S13**（运行中的第一次循环）为 1 重启任务 • 第一个任务循环结束时重置位**%S0** 和**%S13** 到 0 • 重置位**%S31**，**%S38** 和**%S39**（事件控制指示位），和字**%SW48**（事件执行数）。

冷启动过程

冷启动事件中，如果需要一个特殊的应用处理，第一次任务循环中必须检测位**%S1**（为 1）。

电源故障后的输出

一旦检测到电源损耗，输出将被置为（默认）可靠状态（0）。
当电源恢复，输出将处于零状态直到它们被任务再次更新。

控制器初始化

导言

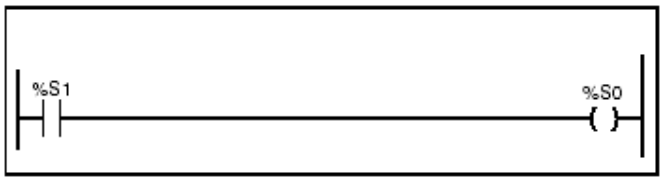
控制器可由 Twido 软件通过设置系统位**%S0**（冷重启）和**%S1**（热重启）来初始化。

冷启动初始化

对于冷启动，系统位**%S0** 必须置为 1。

用**%S0** 和**%S1** 进行热启动初始化

对于热启动初始化，系统位**%S1** 和**%S0** 必须置为 1。
下面示例显示了怎样用系统位对热重启初始化编程。




```
LD %S1
ST %S0
```

注意：不要置%S0 为 1 超过一个控制器扫描。

事件任务管理

5

摘要...

概览

本章描述了事件任务以及它们在控制器中是怎样被执行的。

注意： Twido Brick 10 控制器（TWDLCAA10DRF）不能管理事件任务。

本章包含了什么内容？

本章包含了以下主题：

事件任务概述

导言

前面的章节介绍了周期任务（见周期扫描，p.66）和循环任务（见循环扫描，p.64），对象在这些任务开始和结束时被更新。为使对象能够得到更快的更新，可以利用事件源中断一个确定任务的运行，以执行更高优先级的（事件）任务。

所谓事件任务：

- 是在特定条件（事件源）满足时，所执行的一部分程序，
- 它具有比主程序更高的优先级，
- 它具有更快的响应时间，以使得系统总的响应时间减少。

一个事件的描述

一个事件由以下组成：

- 一个事件源，即用于中断主程序的软件中断或硬件中断条件（见不同事件源的描述，p.81）；
- 一个事件相关的，独立编程的实体；
- 一个事件队列，它用来存储事件列表直至事件被执行；
- 一个优先级，它指定了事件执行的顺序。

不同事件源的描述

不同事件源的概述

一个事件源由特定的软件管理，以保证主程序正确中断，并调用与事件关联的程序。应用程序的扫描时间对事件的执行没有影响。

下面9个事件源是被允许的：

- 4个与VFC函数模块阈值相关联的条件（每个 %VFC场合有两个事件），
- 4个与本地控制器物理输入相关联的条件，
- 1个周期条件。

一个事件源只能对应一个事件，并且必须被TwidoSoft迅速地检测到。一旦被检测到，软件即转到执行与该事件对应的程序部分：以 **SRi:** 为标志的子程序。

本地控制器的物理输入事件

输入%I0.2，%I0.3，%I0.4和%I0.5可以用作事件源，如果它们没被锁定并且这些事件在配置中得到许可。

事件处理可以被本地控制器（位置0）的输入2到输入5在上升沿或下降沿时所激活。

欲知此事件配置更为详尽的信息，请参考“TwidoSoft 操作指导”在线帮助中“硬件配置->输入配置”章节。

%VFC 函数模块的输出事件

%VFC 函数模块的输出 TH0 和 TH1 是事件源。输出 TH0 和 TH1 分别设置为：

- 1，当其电压比阈值S0和S1高时，
- 0，当其电压比阈值S0和S1低时。

这些输出的上升沿或下降沿能激活事件程序。

欲知此事件配置更为详尽的信息，请参考“TwidoSoft 操作指导”在线帮助中“软件配置->超高速计数器”章节。

周期事件	此事件周期性地执行某一程序部分。该任务具有比主任务（主程序）高的优先级。 然而，该事件源的优先级比其它事件源的优先级要低。 该任务的周期在配置中设定，范围5ms到255ms。只能使用一个周期事件。 欲知此事件配置更为详尽的信息，请参考“TwidoSoft 操作指导”在线帮助中“程序参数配置->扫描模式”章节。

事件管理

事件队列和优先级

事件有两种可能的优先级：高和低。但是只有一类事件（即只有一类事件源）具有高优先级。其余事件都是低优先级，它们的执行顺序依赖于他们被检测到的顺序。

为管理事件任务的执行顺序，存在两个事件队列：

- 一个队列里，最多可存储16个高优先级事件（来自同一事件源），
- 另一个队列里，最多可存储16个低优先级事件（来自其它事件源）。

这些队列的管理基于 FIFO 原理：存储在前的事件执行也在前。但是它们只能保存 16 个事件，多余的事件将会被丢失。低优先级队列仅仅在高优先级队列为空时才得到执行。

事件队列管理

每当一个中断发生（检测到事件源）时，下面步骤依次执行：

步骤	描述
1	中断管理： - 物理中断识别， - 事件被存储到对应的事件队列， - 确认没有相同优先级的事件处于未决状态（如果有，该事件将在队列中处于未决状态）。
2	断点保存。
3	对应该事件的程序部分（标记 SRi: 的子程序）执行。
4	输出更新。
5	断点恢复。

在断点恢复之前，队列中所有的事件必须执行完。

事件检查

系统位和字用来检查事件（见系统位和系统字，p.453）：

- %S31：用来执行或延迟一个事件，
- %S38：用来决定是否放置事件到事件队列中，
- %S39：用来查明事件是否未决或丢失，
- %SW48：显示自最近一次冷启动以来有多少事件被执行。

这些位值及字值在冷重启或装载应用程序后将置为零，但是在热重启后保持不变。上述情况下，事件队列都将被置位。

概览

本部分的主题

本部分描述了 Twido 控制器的通信，内置式模拟功能，模拟 I/O 模块管理和 AS-i V2 总线的安装。

本部分包含了哪些内容？

本部分包含了以下章节：

6

概览

本章的主题　　本章提供了 Twido 控制器通信的配置，编程和管理概述。

本章包含了哪些内容？　　本章包含了以下主题：

通信的各种类型介绍

概览

Twido 提供了一个或两个串行通信口用于和远程 I/O 控制器，对等控制器，或普通设备通信。任一端口，如果可用，均可用于任何服务，除了和 Twidosoft 通信，它只能使用第一个端口。每个 Twido 控制器支持三种不同的基本协议：远程连接，ASCII，或 Modbus（Modbus 主协议或 Modbus 从协议）。

远程连接

远程连接协议是一种高速主/从总线。它支持一个主控制器和最多七个从控制器之间的少量数据通信。根据远程控制器的配置，传送相应的应用或 I/O 数据。远程控制器的类型可以是远程 I/O 或对等控制器。

ASCII

ASCII 协议是一个简单的半双工字符模式协议，用于传输和/或接收一个字符串到/自一个简单设备（打印机或中断）。此协议只能通过“EXCH”指令得到支持。

Modbus

Modbus 协议是一个主/从协议，它允许一个并且只能一个主机发送命令，查询从机的响应。主机可单独对一个从机发送命令，也可以广播方式对所有从机发送命令。从机对每一个单独发送给它们的查询返回讯息（响应）。但对广播方式的查询不做响应。

Modbus主机模式－Modbus主机模式允许Twido控制器向从机发出Modbus查询并等待响应。Modbus主机模式只能通过“EXCH”指令得到支持。Modbus ASCII 和 RTU均为Modbus主机模式所支持。

Modbus从机模式－Modbus从机模式允许Twido控制器响应主机的Modbus查询，如果没有配置其它类型的通信，它将是缺省的通信模式。Twido控制器支持供对象访问的标准modbus 数据，控制功能和服务扩展。Modbus ASCII 和 RTU均为Modbus从机模式所支持。

注意：RS-485 网络（没有中继器）可安装 32 个设备（1 个主机和最多 31 个从机），它们的地址可在 1 到 247 之间选择。

Twidosoft 有关控制器通信

概览

每个 Twido 控制器在她的端口 1 上有一个内置的 EIA RS-485 端口。它由内部电源供*电。端口 1*可以用于和 TwidoSoft 编程软件通信。
选件卡或通信模块均不能用于这个连接。不过，调制解调器可以使用这个端口。
下面是 PC 到 Twido 控制器的连接方式：

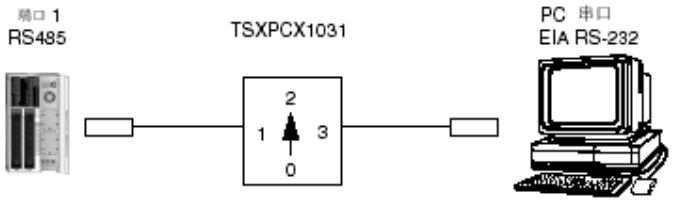
- 通过 TSXPCX 电缆线
- 通过电话线：调制解调器连接

	注意
	设备损坏 当通信电缆 TSXPCX1031 或 TSX PCX 3030 从一个控制器物理移除并很快插入另一个控制器时，TwidoSoft 不能辨识这种断开连接。为了避免这种情况，请在移除电缆之前使用 TwidoSoft 断开连接。 如果不遵守这个警告，将会导致人身伤害或设备损害。

TSXPCX 电缆连接

个人计算机的EIA RS-232C或USB端口通过TSXPCX1031或TSX PCX 3030多功能通信电缆与控制器的端口1相连接。电缆TSX PCX 1031转换EIA RS-232和EIA RS-485间的信号，电缆TSX PCX 3030转换USB和EIA RS-485间的信号。电缆上设有4一位置的旋转开关可供选择不同模式的操作。开关对应的四个位置是“0-3”，TwidoSoft与Twido控制器连接的正确设置是位置2。

连接图如下所示。



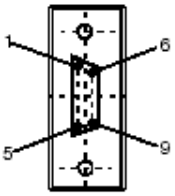
注意：此电缆的 5 号引脚 DPT 信号不等于 0V。这表示控制器的当前连接是 TwidoSoft 连接。对执行固件来说，该内部上拉信号表示这是个 TwidoSoft 连接。

TSXPCX1031 电缆连接器的引脚

下面是 8-引脚 miniDIN 公连接器的引脚图。

Pin outs	RS-485
1	A (+)
2	B (-)
3	NC
4	/DE
5	DPT
6	NC
7	0 V
8	5 V

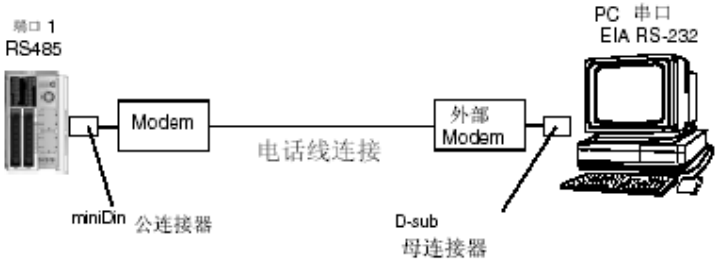
下面是 TSX PCX 1031 的 9-引脚 SubD 母连接器的引脚图。



Pin outs	RS-232
1	DCD
2	RX
3	TX
4	DTR
5	SG
6	NC
7	RTS
8	CTS
9	NC

电话线连接

通过电话线调制解调器连接可以对控制器编程，和控制器通信。与控制器相连的调制解调器是连接控制器端口1的可见调制解调器。与PC相连的可以是内部调制解调器，也可以是连接COM串行口的外部调制解调器。连接图如下所示。



注意：只允许一个调制解调器与控制器的端口 1 相连。

远程连接通信

导言

远程连接协议是一种高速主/从总线。它支持一个主控制器和最多七个远程（从）控制器之间的少量数据通信。根据远程控制器的配置，传送相应的应用或I/O数据。远程控制器的类型可以是远程I/O或对等控制器。

注意：主机包含有关远程 I/O 地址的信息。它不知道地址中的具体控制器。这样，主机不能确认用户程序中用到的远程输入和输出是否实际存在。注意这些远程输入或输出的实际存在。

注意：远程 I/O 总线和协议属于专用，第三方设备不允许出现在网络中。

注意

意外设备操作

- 确信远程连接中只有一个主控制器且每个从机都有唯一地址。如果不遵守这个警告将会导致数据崩溃或意料不到的结果。
- 确信所有从机都有唯一地址。没有两个从机具有相同地址。如果不遵守这个警告将会导致数据崩溃或意料不到的结果。

如果不遵守这个警告，将会导致人身伤害或设备损害。

注意：远程连接需要 EIA RS-485 连接并且一次只能运行一个通信端口。

硬件配置

一个远程连接必须使用最少3-线的EIA RS-485端口。通过配置，可是用第一个端口或第二个端口，如果存在第二个端口的话。

注意：一次只能有一个通信端口配置成远程连接。

下表列出了可以使用的设备：

远程	端口	说明
TWDLCAA10/16/24DRF, TWDLMDA20/40DUK, TWDLMDA20/40DTK, TWDLMDA20DRT	1	基本控制器用miniDIN连接器装置一个3-线EIA RS-485 端口。
TWDNOZ485D	2	通信模块用 miniDIN 连接器装置一个 3-线 EIA RS-485 端口。 注意：该模块只对模块型控制器有用。附加该模块后，控制器不能有操作显示扩展模块。
TWDNOZ485T	2	通信模块用一个终端装置一个 3-线 EIA RS-485 端口。 注意：该模块只对模块型控制器有用。附加该模块后，控制器不能有操作显示扩展模块。
TWDNAC485D	2	通信适配器用 miniDIN 连接器装置一个 3-线 EIA RS-485 端口。 注意：该适配器只对一体型 16 I/O 和 24 I/O 控制器及操作显示扩展模块有用。
TWDNAC485T	2	通信适配器用一个终端装置一个 3-线 EIA RS-485 端口。 注意：该适配器只对一体型 16 I/O 和 24 I/O 控制器及操作显示扩展模块有用。
TWDXCPODM	2	操作显示扩展模块用 miniDIN 连接器装置一个 3-线 EIA RS-485 端口或用一个终端装置一个 3-线 EIA RS-485 端口。 注意：该模块只对模块型控制器有用。附加该模块后，控制器不能有通信扩展模块。

注意：端口 2 的配置（可用性和类型）只在上电时被检查，或被固件可执行程序重置。

每个设备的电缆连接

注意：引脚 5 的 DPT 信号必须与引脚 7 的 0V 相连以表示远程连接通信使用。当此信号不与地相接时，控制器无论主机或从机都默认到模式试图与 TwidoSoft 建立通信。

注意：DPT 与 0V 相连仅为连接到主控制器的端口 1 的必要条件。

每个设备的电缆连接图如下所示。

软件配置

远程连接中只能定义一个主控制器。另外，每个远程控制器必须具有唯一的从机地址。多个主机或从机使用同样地址将会导致传输失败或造成意外。

	注意
	意外设备操作 确信远程连接中只有一个主控制器且每个从机具有唯一地址。如果不遵守这个警告将会导致数据崩溃或意料不到的结果。 如果不遵守这个警告，将会导致人身伤害或设备损害。

主控制器配置

主控制器由TwidoSoft配置，管理最多有七个远程控制器的远程连接网络。这七个远程控制器可配置成远程I/O或对等控制器。由TwidoSoft配置的主机地址对应着地址0。

为了将一个控制器配置成主控制器，需用TwidoSoft将端口1或端口2配置成远程连接且选择地址0（主机）。

然后，从“Add remote controller”窗口，您能指定从控制器为远程I/O或对等控制器以及它们的地址。

远程控制器配置

通过 TwidoSoft 配置端口 1 或 2 为远程连接或分配地址 1 到 7 到端口，完成远程控制器的配置。

下表概括了各种控制器配置的不同和限制：

类型	应用程序	数据访问
远程I/O	没有 甚至没有简单的“END”声明 运行模式与主机相连。	%I 和 %Q 仅控制器的本地I/O可供访问（I/O扩展不可以）。
对等控制器	有 运行模式由主机决定。	%INW 和 %QNW 每个对等控制器可传输一个最多4个字的输入和4个字的输出

远程控制器扫描同步

远程连接的更新循环与主控制器的扫描不同步。与远程控制器的通信由中断驱动且作为与主控制器的扫描运行平行的后台任务发生。扫描循环结束时，最新的数值读进程序数据，被下次程序执行使用。远程I/O和对等控制器有着相同的处理。

所有控制器可通过系统位%S111检查总的连接活动。为了达到同步，主机或对等控制器必须使用系统位%S110。当一个完整的更新循环发生时，这个位被置为1。应用程序负责将它重置为0。

主机可通过系统位%S112使远程连接有效或无效。

控制器可通过%S113检查配置的正确性并更正操作。系统读取并报告端口1上的DPT信号（用于决定TwidoSoft是否被连接）。

概括如下表所示：

系统位	状态	表示
%S100	0	主/从：DPT 非活动的（TwidoSoft 电缆没有连接）
	1	主/从：DPT 活动的（TwidoSoft 电缆连接）
%S110	0	主/从：被程序置为 0
	1	主：所有远程连接交换完成（仅远程 I/O） 从：和主机交换完成
%S111	0	主：一个远程连接交换完成 从：一个远程连接交换被检测
	1	主：一个远程连接交换正在处理 从：一个远程连接交换被检测
%S112	0	主：不能远程连接
	1	主：可以远程连接
%S113	0	主/从：远程连接配置/操作正确
	1	主：远程连接配置/操作错误 从：远程连接操作错误

主控制器重启

如果主控制器重启，将会发生下列某个事件：

- 冷启动(%S0 = 1)强制通信重新初始化。
- 热启动(%S1 = 1) 强制通信重新初始化。
- 在停止模式，主机继续和从机通信。

从控制器重启

如果从控制器重启，将会发生下列某个事件：

- 冷启动(%S0 = 1)强制通信重新初始化。
- 热启动(%S1 = 1) 强制通信重新初始化。
- 在停止模式，从机继续和主机通信。如果主机指示停止状态：
  - 远程 I/O 应用停止状态。
  - 对等控制器继续它的当前状态。

主控制器停止

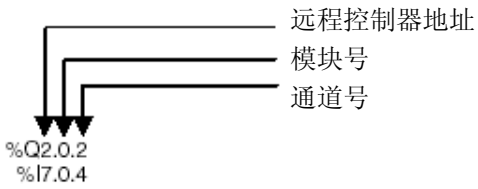
当主控制器进入停止模式，所有的从设备继续与主机通信。当主机指示停止状态时，远程 I/O 将停止，但是对等控制器继续它们当前的运行或停止状态。

远程 I/O 数据访问

配置为远程I/O的远程控制器没有也不执行自己的应用程序。远程控制器本地的数字输入和输出只是主控制器的扩展。应用程序必须也只能使用三位数字的寻址方式。

注意：远程I/O的模块号一般为0。

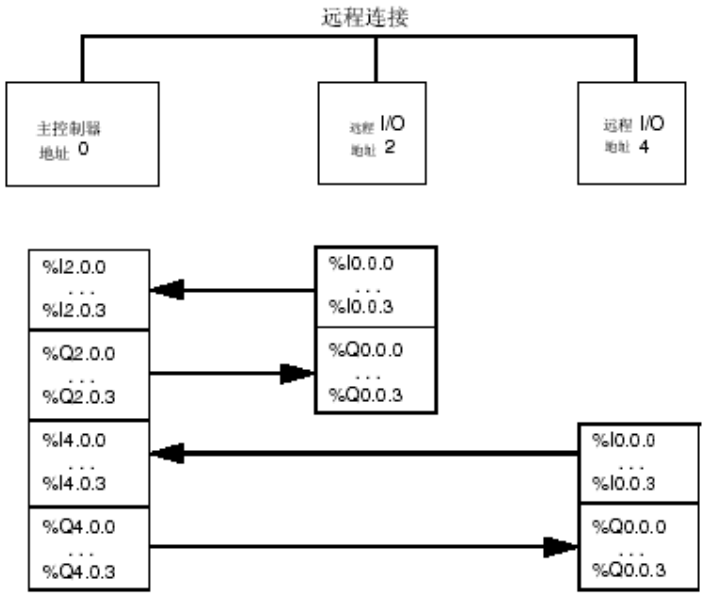
图例



为与远程I/O通信，主控制器使用标准输入和输出符号%I 和 %Q。指令%Q2.0.2可以输出到地址为2的远程I/O的第三个输出位。类似的，指令%I7.0.4为读取7号位置的远程I/O的第五个输入位。

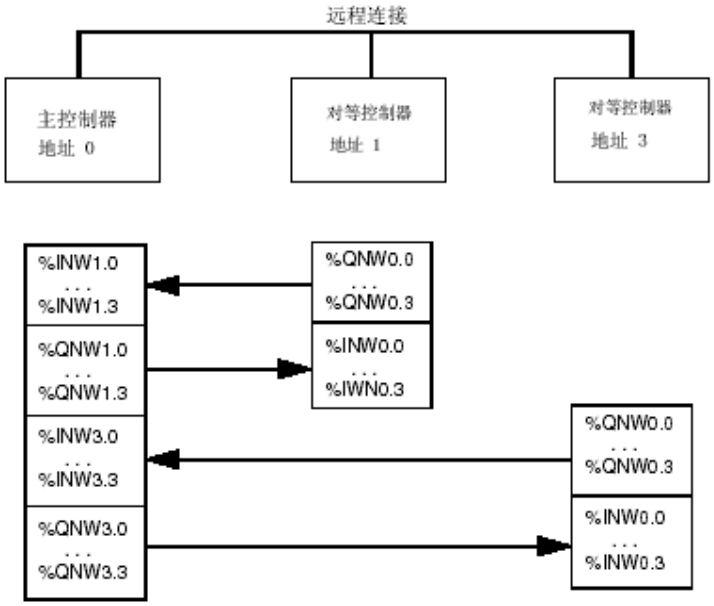
注意：主机限定为只能访问数字I/O，这些I/O只是远程控制器本地I/O的一部分。模拟和扩展I/O不能被传递，除非使用对等通信。

图例



对等控制器数据访问

为与对等控制器通信，主机用网络字%INW和%QNW交换数据。网络中每个对等控制器由其远程地址“j”通过字%INWj.k和%QNWj.k被访问。每个对等控制器使用%INW0.0到%INW0.3和%QNW0.0到%QNW0.3访问主机数据。控制器在运行或停止模式下网络字被自动更新。
下面是一个主机与两个对等控制器数据交换的图例。



远程连接中没有对等消息。主机应用程序可用来管理网络字，为了实现远程控制器间的信息传递，可将主机作为桥梁使用。

状态信息

除了前面解释的系统位，主机还保存当前状态和远程控制器配置。这由系统字%SW111 和 %SW113来完成。远程控制器和主机都能从系统字%SW112获得远程连接通信时最近一次出错值。

系统字	使用	
%SW111	远程连接状态：每个远程控制器两位（仅主机）	
	x0-6	0-远程控制器 1-7 不存在
		1-远程控制器 1-7 存在
	X8-14	0-远程控制器 1-7 检测为远程 I/O
		1-远程控制器 1-7 检测为对等控制器
%SW112	远程连接配置/操作错误码	
		0-操作成功
		1-检测到超时（从机）
		2-检测到校验位出错（从机）
		3-配置不匹配（从机）
%SW113	远程连接配置：每个远程控制器两位（仅主机）	
	x0-6	0-远程控制器 1-7 没有被配置
		1-远程控制器 1-7 已被配置
	X8-14	0-远程控制器 1-7 配置为远程 I/O
		1-远程控制器 1-7 配置为对等控制器

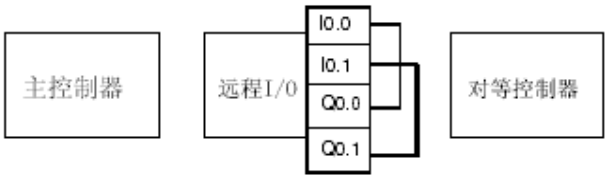
远程连接示例

为配置一个远程连接，您必须：

1. 配置硬件。
2. 连接控制器。
3. 连接PC和控制器间的通信电缆。
4. 配置软件。
5. 写应用程序。

下面是远程I/O与一个对等控制器远程连接使用图例。

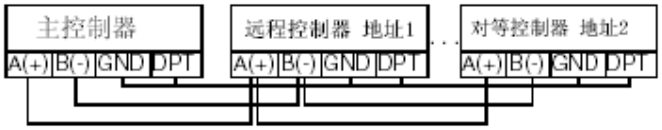
步骤1：配置硬件：



硬件配置是三个任意类型的基本控制器。端口1用于两种通信模式。一种模式通过TwidoSoft配置和传输应用程序。另一种模式用于远程连接网络。如果可行，可以使用控制器的可选端口2，但是一个控制器只支持一个远程连接。

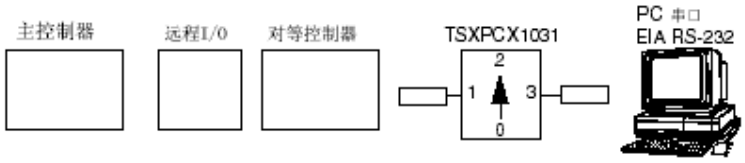
注意：此例中，远程I/O的前两个输入口和前两个输出口用硬连线连接。

步骤1：连接控制器：



将A(+) 和 B(-)信号线分别连在一起。并且每个控制器的DPT信号与地相接。虽然使用端口2（可选卡或通信模块）进行远程连接时不需要此信号与地相接，不过最好养成这个好习惯。

步骤3：连接PC和控制器间的通信电缆：

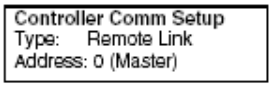


多功能可编程电缆TSXPCX1031或TSX PCX 3030用于三个基本控制器间地互相通信。确信电缆上的开关处于位置2。为了对每个控制器编程，每个控制器都需建立点对点通信。这样建立这种通信：与第一个控制器的端口1连接，传递配置和应用数据，然后将控制器设置成运行状态。对每个控制器重复这个过程。

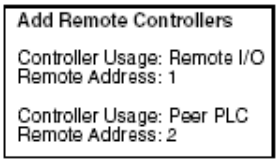
注意：每个控制器在配置和应用传递完毕后需要移除电缆。

步骤4：配置软件：

三个控制器均用TwidoSoft创建配置，和应用程序，如果可以的话。
对于主控制器，在控制器通信安装编辑中设置协议为“远程连接”，地址为“0（主机）”。



主机通过添加“远程I/O”地址“1”和“对等PLC”地址“2”配置远程控制器。



对于配置成远程I/O的控制器，确认控制器通信安装设置成“远程连接”且其地址设为“1”。

```
Controller Comm Setup
Type:    Remote Link
Address: 1
```

对于配置成对等机的控制器，确认控制器通信安装设置成“远程连接”且其地址设为“2”。

```
Controller Comm Setup
Type:    Remote Link
Address: 2
```

步骤5：写应用程序：

对主控制器，写如下应用程序：

```
LD 1

[%MW0 := %MW0 +1]
[%QNW2.0 := %MW0]
[%MW1 := %INW2.0]

LD %I0.0
ST %Q1.0.0
LD %I1.0.0
ST %Q0.0

LD %I0.1
ST %Q1.0.1
LD %I1.0.1
ST %Q0.1
```

对配置成远程I/O的控制器，不用写任何应用程序。
对配置成对等机的控制器，写如下应用程序：

```
LD 1
[%QNW0.0 := %INW0.0]
```

此例中，主机程序增加一个内部存储字且用一个网络字传递给对等控制器。对等控制器从主机接收该字并将其返回。主机将接收一个不同的存贮字并保存这次传输。
为与远程I/O控制器通信，主机将自己本地输入传送给远程I/O输出。当远程I/O外部I/O硬连线时，主机可返回并重新得到信号。

导言

ASCII协议为Twido控制器提供了一个的半双工字符模式协议，用于和一个设备传输和/或接收数据。该协议由EXCHx指令支持，由%MSGx功能模块控制。

ASCII协议提供了三种通信方式：

- 只发送
- 发送/接收
- 只接收

EXCHx指令发送和/或接收帧的最大值是128字节。

硬件配置

ASCII连接可以通过EIA RS-232或EIA RS-485端口建立，并且可以同时在两个通信端口上运行。

下表列出了可以使用的设备：

远程	端口	说明
TWDLCAA10/16/24DRF, TWDLMDA20/40DUK, TWDLMDA20/40DTK, TWDLMDA20DRT	1	基本控制器用miniDIN连接器装置一个3-线EIA RS-485 端口。
TWDNOZ232D	2	通信模块用 miniDIN 连接器装置一个 3-线 EIA RS-232 端口。 注意：该模块只对模块型控制器有用。附加该模块后，控制器不能有操作显示扩展模块。
TWDNOZ485D	2	通信模块用 miniDIN 连接器装置一个 3-线 EIA RS-485 端口。 注意：该模块只对模块型控制器有用。附加该模块后，控制器不能有操作显示扩展模块。
TWDNOZ485T	2	通信模块用一个终端装置一个 3-线 EIA RS-485 端口。 注意：该模块只对模块型控制器有用。附加该模块后，控制器不能有操作显示扩展模块。
TWDNAC232D	2	通信适配器用 miniDIN 连接器装置一个 3-线 EIA RS-232 端口。 注意：该适配器只对一体型 16 I/O 和 24 I/O 控制器及操作显示扩展模块有用。
TWDNAC485D	2	通信适配器用 miniDIN 连接器装置一个 3-线 EIA RS-485 端口。 注意：该适配器只对一体型 16 I/O 和 24 I/O 控制器及操作显示扩展模块有用。
TWDNAC485T	2	通信适配器用一个终端装置一个 3-线 EIA RS-485 端口。 注意：该适配器只对一体型 16 I/O 和 24 I/O 控制器及操作显示扩展模块有用。

TWDXCPODM	2	操作显示扩展模块用 miniDIN 连接器装置一个 3-线 EIA RS-232 端口，用 miniDIN 连接器装置一个 3-线 EIA RS-485 端口或用一个终端装置一个 3-线 EIA RS-485 端口。 注意：该模块只对模块型控制器有用。附加该模块后，控制器不能有通信扩展模块。

注意：端口 2 的配置（可用性和类型）只在上电时被检查，或被固件可执行程序重置。

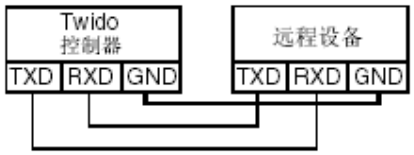
定义电缆

下面是 EIA RS-232 和 EIA RS-485 型的电缆连接定义图。

注意：如果 Twido 控制器使用端口 1，5 号引脚的 DPT 信号必须与 7 号引脚的 0V 相接。这意味着 Twido 控制器的端口 1 通信是 ASCII 而不是和 TwidoSoft 软件通信。

每个设备的电缆连接图如下。

EIA RS-232 电缆



EIA RS-485 电缆

软件配置

为了配置控制器使用ASCII协议通过串行接口发送和接收字符，您必须：

步骤	描述
1	用 TwidoSoft 配置 ASCII 串行口。
2	在应用程序中创建发送/接收表以供 EXCHx 指令使用。

端口配置

Twido控制器可用基本端口1或可选端口2来使用ASCII协议。配置ASCII串行口：

步骤	动作
1	定义本地控制器配置的附加通信适配器或模块。
2	右键点击端口再点击控制器通信安装编辑…并将串口类型改为“ASCII”。
3	设置相关通信参数。

ASCII 模式发送/接收表配置

发送和/或接收帧的最大值是 128 字节。与 EXCHx 指令相关的字表由发送和接收控制表组成。

	高字节	低字节
控制表	命令	长度（发送/接收）
	保留（0）	保留（0）
发送表	发送字节 1	发送字节 2
	…	…
	…	发送字节 n
	发送字节 n+1	
接收表	接收字节 1	接收字节 2
	…	…
	…	接收字节 p
	接收字节 p+1	

控制表

长度字节包含发送表的长度，如果接收被请求，它将被接收结束时收到的字符数覆盖。

命令字节必须包含下面之一：

- 0：只发送
- 1：发送/接收
- 2：只接收

发送/接收表

在只发送模式，控制和发送表在 EXCHx 指令执行之前被填写，类型可以是%KW 或%MW。只发送模式不需要接收字符空间。一旦所有字节发送完毕，%MSGx.D 将置为 1，然后可以执行新的 EXCHx 指令。

在发送/接收模式，控制和发送表在EXCHx指令执行之前被填写，并且类型必须是%MW。发送表的末端需要最多128接收字节的空间。一旦所有字节发送完毕，Twido控制器转换到接收模式并等待接收任何字节。

在只接收模式，控制表在EXCHx指令执行之前被填写，并且类型必须是%MW。控制表的末端需要最多128接收字节的空间。Twido控制器立即进入接收模式并等待接收任何字节。

当收到帧结束字节或接收表满时接收结束。如果配置了一个非零停止时间，接收在停止时间到时也将结束。如果停止时间值选为零，则没有接收停止时间。这样为了停止接收，必须激活%MSGx.R输入。

消息交换

语言提供了两种通信服务：

- **EXCHx instruction:** 发送/接收消息
- **%MSGx Function Block:** 控制消息交换

Twido 控制器处理 EXCHx 指令时使用端口配置协议。

注意：每个通信端口可配置不同或相同协议。通过设置端口号（1 或 2）路径 EXCHx 指令和%MSGx 功能模块可以访问每个通信端口。

EXCHx 指令

EXCHx指令允许Twido控制器发送和/或接收信息到/自ASCII设备。用户定义一个字表（%MWi:L或%KWi:L），其中包含用来发送和/或接收的控制信息和数据（发送和/或接收最多128字节）。字表格式如前描述。使用EXCHx指令完成消息交换：

语法：[EXCHx %MWi:L] 或 [EXCHx %KWi:L]
这里：x＝端口号（1或2）
L＝控制字以及发送和接收表中的字数

Twido 控制器必须在第二条 EXCHx 指令执行之前由第一条指令完成交换。发送不止一条消息时，必须使用%MSGx 功能模块。
当任何传输处于中断控制情形时（数据接收也处于中断控制情形），EXCHx列表指令的处理立即开始，并被视为后台处理。

%MSGx 功能模块

%MSGx功能模块的使用不是必需的。它能用于管理数据交换。%MSGx功能模块有三种用途：

- 通信错误校验

错误校验确认EXCHx指令编程参数L（字表长度）足够大，能包含发送消息的长度。与存储在字表第一个字的低字节中的长度相比较。

- 多消息协调

%MSGx功能模块提供前面消息传输完成的时间信息，以保证多消息发送的协调。

- 传输优先消息

%MSGx功能模块允许当前消息传输停止以发送紧急消息。

%MSGx功能模块有一个输入，两个输出：

输入/输出	定义	描述
R	输入重置	置为1：通信重新初始化或模块重置（%MSGx.E = 0和%MSGx.D =1）。
%MSGx.D	通信完成	0：程序请求。 1：通信完成，如果：传输完毕，字符接收完毕，出错，或模块重置。
%MSGx.E	错误	0：消息长度正确且连接正确。 1：如果命令错误，表配置错误，接收字符错误（速率，奇偶校验，等等。），或接收表满。

限制

请务必注意下列限制：

- 端口 2 的可用性和类型（见%SW7）只在上电或重置时被检查
- 当连接 TwidoSoft 时端口 1 将不能进行任何消息处理
- EXCHx 和%MSG 不能被配置为远程连接的端口所处理
- EXCHx 将停止 Modbus 从机处理
- EXCHx 指令的处理在错误事件中不会得到重试
- 输入重置(R)可用来中断 EXCHx 指令接收处理
- EXCHx 指令可配置停止时间来中断接收
- 多消息通过%MSGx.D 得到控制

错误和工作模式环境

当使用EXCHx指令时如果出错，位%MSGx.D和%MSGx.E将置为1且系统字%SW63包含端口1的错误代码，%SW64包含端口2的错误代码。

系统字	用法
%SW63	EXCH1 错误代码： 0 – 操作成功 1 – 传输字节数过大 (> 128) 2 – 发送表太小 3 – 字表太小 4 – 接收表溢出 5 – 停止时间到 6 – 发送错误 7 – 表中错误命令 8 – 所选端口没有配置/不可用 9 – 接收错误 (仅 ASCII 模式) 10 – 接收时不能用 %KW 11 – 发送偏移量大于发送表 12 – 接收偏移量大于接收表 13 – 控制器停止 EXCH 处理
%SW64	EXCH2 错误代码：见%SW63。

通信过程中控制器重启结果

如果控制器重启，下面事件中的一个将会发生：

- 冷启动(%S0 = 1)强制通信重新初始化。
- 热启动(%S1 = 1)强制通信重新初始化。
- 在停止模式，控制器停止所有 ASCII 通信。

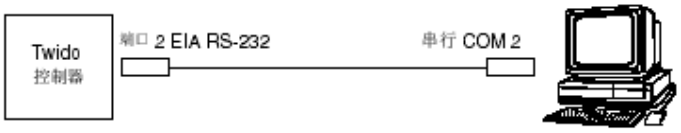
ASCII 连接示例

为配置一个ASCII连接，您必须：

1. 配置硬件。
2. 连接ASCII通信电缆。
3. 配置端口。
4. 写应用程序。
5. 初始化活动表编辑器。

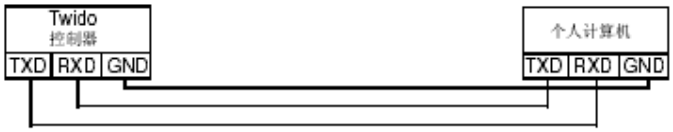
下面图描述了怎样使用 ASCII 和 PC 终端仿真器通信。

步骤 1：配置硬件：



硬件配置是指两个串行连接，从PC到Twido控制器的可选EIA RS-232端口2。对于模块型控制器，可选端口2是指TWDXCPODM上的TWDNOZ232D或TWDNAC232D。上图模块型控制器，可选端口2是TWDNAC232D。为了配置控制器，需连接电缆TSXPCX1031（图中没有显示）至Twido控制器的端口1。然后连接电缆至PC的COM 1口。确信电缆开关处于位置2。最后，连接PC的COM 2口和Twido控制器的可选EIA RS-232端口2。连线说明见下一步。

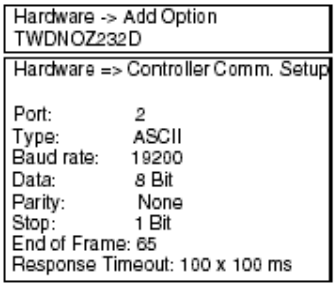
步骤 2：ASCII 通信电缆（EIA RS-232）连线说明：

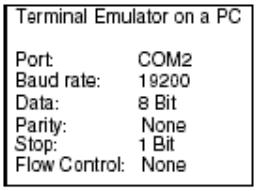


ASCII通信电缆的最少使用线数是3。发送和接收信号线交叉。

注意：电缆的 PC 端，可能需要额外的连接（如数据终端准备和数据设置准备）来满足握手信号。Twido 控制器不需要额外连接。

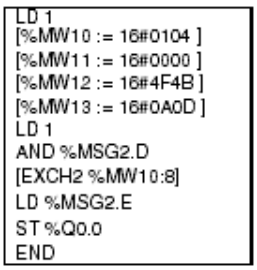
步骤 3：端口配置：





用一个PC终端仿真程序配置COM2口并确保不存在流控制。
用TwidoSoft配置控制器的端口。首先，配置硬件选项。此例中，TWDNOZ232D加到模块型基本控制器。
其次，用PC终端仿真器的所有相同参数设置初始化控制器通信安装。此例中，选择大写字母“A”表示“帧结束”，用于停止字符接收。“响应停止时间”参数选用10秒。这两个参数只有一个被调用，取决于哪个先发生。

步骤4：写应用程序：



用TwidoSoft创建一个应用程序包括三个主要部分。首先，初始化控制和发送表以便EXCH指令使用。此例中，设置命令发送和接收数据。数据发送数量设为4字节并初始化为字符："O", "K", CR, LF。

然后，检查相关通信位%MSG2，如果端口准备好就发送EXCH2指令。EXCH2指令指定了一个8字值。它们是2个控制字(%MW10 and %MW11)，2个字用于传输信息(%MW12 and %MW13)，4个字接收数据(%MW14到%MW17)。
最后，错误状态%MSG2被得到并存储于本地基本控制器I/O的第一个输出位。还可以使用%SW64进行附加的错误检查以使结果更加准确。

步骤 5：初始化活动表编辑器：

Address	Current	Retained	Format
1 %MW10	0104	0000	Hexadecimal
2 %MW11	0000	0000	Hexadecimal
3 %MW12	4F4B	0000	Hexadecimal
4 %MW13	0A0D	0000	Hexadecimal
5 %MW14	TW	0000	ASCII
6 %MW15	ID	0000	ASCII
7 %MW16	O	0000	ASCII
8 %MW17	A	0000	ASCII

最后一步是下载应用程序到控制器并运行它。初始化活动表编辑器以激活和显示字%MW10到%MW17。字符"O"-"K"-CR-LF被显示到终端仿真器上。当EXCH模块的响应时间到且新EXCH模块没有开始之前，字符"O"-"K"-CR-LF可被多次显示。终端仿真器敲下"T"-"W"-"I"-"D"-"O"-" "-"A"。它们将被交换到Twido控制器并被活动表编辑器所显示。

Modbus 通信

导言

Modbus 协议是一个主-从协议，它允许一个并且只能一个主机发送命令，查询从机的响应。主机可单独对一个从机发送命令，也可以广播方式对所有从机发送命令。从机对每一个单独发送给它们的查询返回讯息（响应）。但对广播方式的查询不做响应。

硬件配置

Modbus连接可以通过EIA RS-232或EIA RS-485端口建立，并且可以同时在两个通信端口上运行。每个端口可指定为自己的Modbus地址。

下表列出了可以使用的设备：

远程	端口	说明
TWDLCAA10/16/24DRF, TWDLMDA20/40DUK, TWDLMDA20/40DTK, TWDLMDA20DRT	1	基本控制器用miniDIN连接器装置一个3-线EIA RS-485 端口。
TWDNOZ232D	2	通信模块用 miniDIN 连接器装置一个 3-线 EIA RS-232 端口。 注意：该模块只对模块型控制器有用。附加该模块后，控制器不能有操作显示扩展模块。
TWDNOZ485D	2	通信模块用 miniDIN 连接器装置一个 3-线 EIA RS-485 端口。 注意：该模块只对模块型控制器有用。附加该模块后，控制器不能有操作显示扩展模块。
TWDNOZ485T	2	通信模块用一个终端装置一个 3-线 EIA RS-485 端口。 注意：该模块只对模块型控制器有用。附加该模块后，控制器不能有操作显示扩展模块。
TWDNAC232D	2	通信适配器用 miniDIN 连接器装置一个 3-线 EIA RS-232 端口。 注意：该适配器只对一体型 16 I/O 和 24 I/O 控制器及操作显示扩展模块有用。
TWDNAC485D	2	通信适配器用 miniDIN 连接器装置一个 3-线 EIA RS-485 端口。 注意：该适配器只对一体型 16 I/O 和 24 I/O 控制器及操作显示扩展模块有用。
TWDNAC485T	2	通信适配器用一个终端装置一个 3-线 EIA RS-485 端口。 注意：该适配器只对一体型 16 I/O 和 24 I/O 控制器及操作显示扩展模块有用。

TWDXCPODM	2	操作显示扩展模块用 miniDIN 连接器装置一个 3-线 EIA RS-232 端口，用 miniDIN 连接器装置一个 3-线 EIA RS-485 端口或用一个终端装置一个 3-线 EIA RS-485 端口。 注意：该模块只对模块型控制器有用。附加该模块后，控制器不能有通信扩展模块。

注意：端口 2 的配置（可用性和类型）只在上电时被检查，或被固件可执行程序重置。

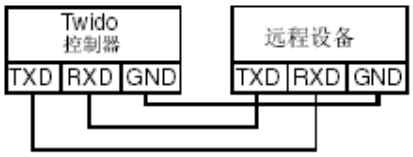
定义电缆

下面是 EIA RS-232 和 EIA RS-485 型的电缆连接定义图。

注意：如果 Twido 控制器使用端口 1，5 号引脚的 DPT 信号必须与 7 号引脚的 0V 相接。这意味着 Twido 控制器的端口 1 通信是 Modbus 而不是和 TwidoSoft 软件通信。

每个设备的电缆连接图如下。

EIA RS-232 电缆



EIA RS-485 电缆

软件配置

为了配置控制器使用Modbus协议通过串行接口发送和接收字符，您必须：

步骤	描述
1	用 TwidoSoft 配置 ASCII 串行口。
2	在应用程序中创建发送/接收表以供 EXCHx 指令使用。

端口配置

Twido 控制器可用基本端口 1 或可选端口 2 来使用 Modbus 协议。配置 Modbus 串行口：

步骤	动作
1	定义本地控制器配置的附加通信适配器或模块。
2	右键点击端口再点击控制器通信安装编辑…并将串口类型改为“Modbus”。
3	设置相关通信参数。

Modbus 主模式

Modbus 主模式允许控制器向从机发送一个 Modbus 查询，并等待其响应。Modbus 主模式只能通过 EXCHx 指令得到支持。Modbus ASCII 和 RTU 均被 Modbus 主模式支持。

发送和/或接收帧的最大值是 128 字节。

另外与 EXCHx 指令相关的字表由控制，发送和接收表组成。

	高字节	低字节
控制表	命令	长度（发送/接收）
	接收偏移	发送偏移
发送表	发送字节 1	发送字节 2
	…	…
	…	发送字节 n
	发送字节 n+1	
接收表	接收字节 1	接收字节 2
	…	…
	…	接收字节 p
	接收字节 p+1	

控制表	长度字节包含发送表的长度，如果接收被请求，它将被接收结束时收到的字符数覆盖。 该参数是发送表的字节长度。如果Tx偏移参数等于0，该参数将等于发送帧的长度。如果Tx偏移参数不等于0，发送表的一个字节（由偏移值决定）将不被发送且该参数等于帧长度加1。 命令字节在Modbus RTU查询（除了广播）情形下必须总是等于1（Tx和Rx）。 Tx偏移字节包含字节发送时被忽略的字节在发送表中的排列号（1表示第一个字节，2表示第二个字节，等等）。它用于处理Modbus协议中与字节/字的值有关的问题。例如，如果此字节包含3，则第三个字节将被忽略，使得表中第四个字节在发送时变为第三个字节。 Rx偏移字节包含信息包发送时加入的字节在接收表中的排列号（1表示第一个字节，2表示第二个字节，等等）。它用于处理Modbus协议中与字节/字的值有关的问题。例如，如果此字节包含3，则表中第三个字节将被填为零，使得实际接收到的第三个字节在表中变为第四个字节。

发送/接收表

在任一模式(Modbus ASCII 或 Modbus RTU)，发送表在EXCHx指令执行之前被填写。在执行时间，控制器决定什么是数据链路层，并完成所有必要的转换以处理传输和响应。发送/接收表不存储开始，结束和检查字符。

一旦所有字节发送完毕，控制器转换到接收模式并等待接收任何字节。接收通过下面某种方式完成：

- 字符或帧的停止时间被检测到，
- ASCII模式中收到帧结束字符，
- 接收表满。

Received Byte X条目包含Modbus协议（RTU编码）数据，这些数据将被发送。如果通信端口配置成Modbus ASCII，发送时将附加合适的帧字符。第一个字节包含设备地址（特殊或广播），第二个字节包含功能代码，剩下的字节包含功能代码相关的信息。

注意：这是一个典型应用，但没有定义所有可能性。数据发送时将不进行确认工作。

Received Byte X 条目包含 Modbus 协议（RTU 编码）数据，这些数据将被接收。如果通信端口配置成 Modbus ASCII，响应时将移除对应的帧字符。第一个字节包含设备地址，第二个字节包含功能代码（或响应代码），剩下的字节包含功能代码相关的信息。

注意：这是一个典型应用，但没有定义所有可能性。数据接收时除了校验，将不进行别的确认工作。

Modbus 从模式

Modbus从模式允许控制器响应Modbus主机的标准Modbus查询。
当电缆TSXPCX1031与控制器相连时，端口开始TwidoSoft通信，电缆连接之前所运行的通信模式将被临时停止。
Modbus协议支持两种数据链路层格式：ASCII和RTU。每种格式都由物理层定义，ASCII使用7个数据位，RTU使用8个数据位。
当使用Modbus ASCII模式时，消息的每个字节作为两个ASCII发送。Modbus ASCII帧从一个起始字符(':')开始，可用两个终止字符(CR and LF)表示结束。帧结束字符默认为0x0A（换行），用户可在配置中修改这个字节。Modbus ASCII帧的校验值是除去起始和终止字符后帧的二进制补码。
Modbus RTU模式在消息发送之前不重新定义格式；然而，它使用一个不同的校验计算模式CRC。
Modbus数据链路层有下列限制：

- 地址1-247
- 位：请求可有128位
- 字：请求可有64个16位的字

消息交换

语言提供了两种通信服务：

- **EXCHx instruction:** 发送/接收消息
- **%MSGx Function Block:** 控制消息交换

Twido 控制器处理 EXCHx 指令时使用端口配置协议。

注意：每个通信端口可配置不同或相同协议。通过设置端口号（1 或 2）路径 EXCHx 指令和%MSGx 功能模块可以访问每个通信端口。

EXCHx 指令

EXCHx指令允许Twido控制器发送和/或接收信息到/自Modbus设备。用户定义一个字表（%MWi:L或%KWi:L），其中包含用来发送和/或接收的控制信息和数据（发送和/或接收最多128字节）。字表格式如前描述。使用EXCHx指令完成消息交换：

语法：[EXCHx %MWi:L] 或 [EXCHx %KWi:L]
这里：x＝端口号（1或2）
L＝控制字以及发送和接收表中的字数

Twido 控制器必须在第二条 EXCHx 指令执行之前由第一条指令完成交换。发送不止一条消息时，必须使用%MSGx 功能模块。
当任何传输处于中断控制情形时（数据接收也处于中断控制情形），EXCHx列表指令的处理立即开始，并被视为后台处理。

%MSGx 功能模块

%MSGx功能模块的使用不是必需的。它能用于管理数据交换。%MSGx功能模块有三种用途：

- 通信错误校验

错误校验确认EXCHx指令编程参数L（字表长度）足够大，能包含发送消息的长度。与存储在字表第一个字的低字节中的长度相比较。

- 多消息协调

%MSGx功能模块提供前面消息传输完成的时间信息，以保证多消息发送的协调。

- 传输优先消息

%MSGx功能模块允许当前消息传输停止以发送紧急消息。

%MSGx功能模块有一个输入，两个输出：

输入/输出	定义	描述
R	输入重置	置为1：通信重新初始化或模块重置（%MSGx.E = 0和%MSGx.D =1）。
%MSGx.D	通信完成	0：程序请求。 1：通信完成，如果：传输完毕，字符接收完毕，出错，或模块重置。
%MSGx.E	错误	0：消息长度正确且连接正确。 1：如果命令错误，表配置错误，接收字符错误（速率，奇偶校验，等等。），或接收表满。

限制

请务必注意下列限制：

- 端口 2 的可用性和类型（见%SW7）只在上电或重置时被检查
- 当连接 TwidoSoft 时端口 1 将不能进行任何消息处理
- EXCHx 和%MSG 不能被配置为远程连接的端口所处理
- EXCHx 将停止 Modbus 从机处理
- EXCHx 指令的处理在错误事件中不会得到重试
- 输入重置(R)可用来中断 EXCHx 指令接收处理
- EXCHx 指令可配置停止时间来中断接收
- 多消息通过%MSGx.D 得到控制

错误和工作模式环境

当使用EXCHx指令时如果出错，位%MSGx.D和%MSGx.E将置为1且系统字%SW63包含端口1的错误代码，%SW64包含端口2的错误代码。

系统字	用法
%SW63	EXCH1 错误代码： 0 – 操作成功 1 – 传输字节数过大 (> 128) 2 – 发送表太小 3 – 字表太小 4 – 接收表溢出 5 – 停止时间到 6 – 发送错误 7 – 表中错误命令 8 – 所选端口没有配置/不可用 9 – 接收错误 (仅 ASCII 模式) 10 – 接收时不能用 %KW 11 – 发送偏移量大于发送表 12 – 接收偏移量大于接收表 13 – 控制器停止 EXCH 处理
%SW64	EXCH2 错误代码：见%SW63。

主控制器重启

如果主/从控制器重启，下面事件中的一个将会发生：

- 冷启动(%S0 = 1)强制通信重新初始化。
- 热启动(%S1 = 1)强制通信重新初始化。
- 在停止模式，控制器停止所有 Modbus 通信。

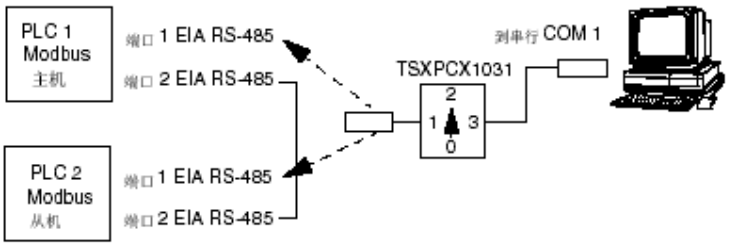
Modbus 连接示例 1

为配置一个Modbus连接，您必须：

1. 配置硬件。
2. 连接Modbus通信电缆。
3. 配置端口。
4. 写应用程序。
5. 初始化活动表编辑器。

下面图描述了怎样使用 Modbus 请求代码 3 读取一个从机的输出字。此例使用两个 Twido 控制器。

步骤 1：配置硬件：



硬件配置是两个Twido控制器。一个配置成Modbus主机，另一个配置成Modbus从机。

注意：此例中，每个控制器配置成使用 EIA RS-485 端口 1 和可选 EIA RS-485 端口 2。模块型控制器的端口 2 可以是 TWDNOZ485D 或 TWDNOZ485T，如果您使用 TWDXCPODM，它可以是 TWDNAC485D 或 TWDNAC485T。一体型控制器的端口 2 可以是 TWDNAC485D 或 TWDNAC485T。

为了配置每个控制器，需连接电缆TSXPCX1031至控制器的端口1。

注意：电缆 TSXPCX1031 一次只能连接到一个控制器，并且只能连接到 EIA RS-485 端口 1。

然后连接电缆至PC的COM 1口。确信电缆开关处于位置2。下载和监控应用程序。对第二个控制器重复此过程。

步骤 2：连接 Modbus 通信电缆：

此例中连线是点对点连接。三个信号 A(+), B(-), 和 GND 连线如图所示。如果使用 Twido 控制器的端口 1，DPT 信号（引脚 5）必须与 0V（引脚 7）相接。DPT 的这个条件决定 TwidoSoft 是否被连接。当与地相接时，控制器将使用程序中的端口配置设置决定通信方式。

步骤 3：端口配置：

Hardware -> Add Option TWDNOZ485-	
Hardware => Controller Comm. Setup	
Port:	2
Type:	Modbus
Address:	1
Baud rate:	19200
Data:	8 bits
Parity:	None
Stop:	1 bits
End of frame:	65
Response Timeout:	10 x 100 ms
Frame overflow:	10 ms

Hardware -> Add Option TWDNOZ485-	
Hardware => Controller Comm. Setup	
Port:	2
Type:	Modbus
Address:	2
Baud rate:	19200
Data:	8 bits
Parity:	None
Stop:	1 bits
End of frame:	65
Response Timeout:	100 x 100 ms
Frame overflow:	10 ms

在所有主机和从机应用中，可选 EIA RS-485 端口将被配置。确信控制器通信参数在 Modbus 协议和不同地址中得到改正。
此例中，主机地址设为1，从机地址设为2。位数设为8，表示我们将使用Modbus RTU模式。如果它被设为7，则表示使用Modbus-ASCII模式。其它的默认修改是将响应的停止时间增加到1秒。

注意：因为选择的是 Modbus RTU 模式，所以“帧结束”参数被忽略。

步骤 4：写应用程序：

```
LD 1
[%MW0 := 16#0106]
[%MW1 := 16#0300]
[%MW2 := 16#0203]
[%MW3 := 16#0000]
[%MW4 := 16#0004]
LD 1
AND %MSG2.D
[EXCH2 %MW0:11]
LD %MSG2.E
ST %Q0.0
END
```

```
LD 1
[%MW0 := 16#6566]
[%MW1 := 16#6768]
[%MW2 := 16#6970]
[%MW3 := 16#7172]
END
```

使用 TwidoSoft 为主机和从机写应用程序。对于从机，我们简单写一些存储字到一个已知值集合。主机中，EXCHx 指令的字表初始化为从 Modbus 地址为 2 的从机读取 4 个字，从位置%MW0 开始。

注意：注意 Modbus 主机的%MW1 中 RX 偏移设置的使用。偏移 3 将在表的接收区域的第三个位置加入一个字节（值= 0）。主机中字的这种排列使得它们正确设置字的边界。如果没有这个偏移，每个数据字将被交换模块的两个字分开。偏移使得用法方便。

在执行EXCH2之前，程序检查和%MSG2相关的通信位。最后，%MSG2的错误状态被得到并存储于本地基本控制器I/O的第一个输出位。另外还可以使用%SW64进行错误检查，以便结果更为准确。

步骤 5：初始化主机的活动表编辑器：

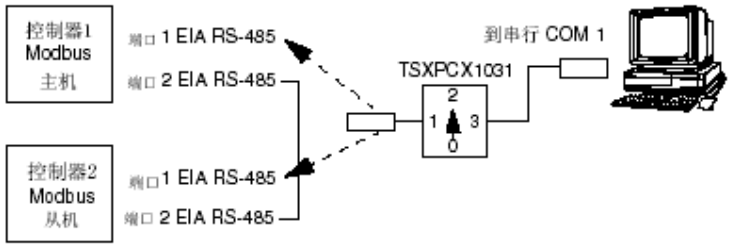
	Address	Current	Retained	Format
1	%MW5	0203	0000	Hexadecimal
2	%MW6	0008	0000	Hexadecimal
3	%MW7	6566	0000	Hexadecimal
4	%MW8	6768	0000	Hexadecimal
5	%MW9	6970	0000	Hexadecimal
6	%MW10	7172	0000	Hexadecimal

在下载并设置每个控制器运行之后，打开主机的活动表。检查表的响应部分，确认响应代码是 3 且被读字节数正确。此例中，注意从从机读取的字（开始于%MW7）在主机中排列正确，包括字的边界。

Modbus 连接示例 2

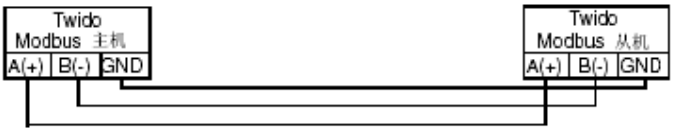
下面图描述了怎样使用 Modbus 请求代码 16 给一个从机写输出字。此例使用两个 Twido 控制器。

步骤 1：配置硬件：



硬件配置与上例相同。

步骤 2：连接 Modbus 通信电缆(RS-485)：



Modbus 通信电缆与上例相同。

步骤 3：端口配置：

Hardware -> Add Option TWDNOZ485-
Hardware => Controller Comm. Setup
Port: 2
Type: Modbus
Address: 1
Baud rate: 19200
Data: 8 Bit
Parity: None
Stop: 1 Bit
End of Frame: 65
Response Timeout: 10 x 100 ms
Frame Timeout: 10 ms

Hardware -> Add Option TWDNOZ485-
Hardware => Controller Comm. Setup
Port: 2
Type: Modbus
Address: 2
Baud rate: 19200
Data: 8 Bit
Parity: None
Stop: 1 Bit
End of Frame: 65
Response Timeout: 100 x 100 ms
Frame Timeout: 10 ms

端口配置与上例相同。

步骤 4：写应用程序：

```
LD 1
[%MW0 := 16#010C ]
[%MW1 := 16#0007 ]
[%MW2 := 16#0210 ]
[%MW3 := 16#0010 ]
[%MW4 := 16#0002 ]
[%MW5 := 16#0004 ]
[%MW6 := 16#6566 ]
[%MW7 := 16#6768 ]
LD 1
AND %MSG2.D
[EXCH2 %MW0:11]
LD %MSG2.E
ST %Q0.0
END
```

```
LD 1
[%MW18 := 16#FFFF ]
END
```

使用 TwidoSoft 为主机和从机写应用程序。对于从机，写一个简单存储字%MW18。它将给从机分配空间，存储地址%MW0 到%MW18。如果没有分配这个空间，Modbus 请求将试图写到从机中不存在的位置。主机中，EXCHx 指令的字表初始化为读 4 个字节到 Modbus 地址为 2 的从机，地址是%MW16（十六进制 10）。

注意：注意 Modbus 主机程序的%MW1 中发送偏移设置的使用。偏移 7 将使第六个字的高字节（%MW5 中十六进制值 00）禁止。这样使得字表的发送表中数据值排列在一起，使得它们在数据分界处正确分开。

在执行EXCH2之前，程序检查和%MSG2相关的通信位。最后，%MSG2的错误状态被得到并存储于本地基本控制器I/O的第一个输出位。另外还可以使用%SW64进行错误检查，以便结果更为准确。

步骤 5：初始化活动表编辑器：

创建主机的活动表如下：

	Address	Current	Retained	Format
1	%MW0	010C	0000	Hexadecimal
2	%MW1	0007	0000	Hexadecimal
3	%MW2	0210	0000	Hexadecimal
4	%MW3	0010	0000	Hexadecimal
5	%MW4	0002	0000	Hexadecimal
6	%MW5	0004	0000	Hexadecimal
7	%MW6	6566	0000	Hexadecimal
8	%MW7	6768	0000	Hexadecimal
9	%MW8	0210	0000	Hexadecimal
10	%MW9	0010	0000	Hexadecimal
11	%MW10	0004	0000	Hexadecimal

创建从机的活动表如下：

	Address	Current	Retained	Format
1	%MW16	6566	0000	Hexadecimal
2	%MW17	6768	0000	Hexadecimal

在下载并设置每个控制器运行之后，打开从控制器的活动表。%MW16 和 %MW17 的两个值被写入从机。主机中，活动表用于检查交换数据的接收表部分。该数据显示上述示例中的从机地址，响应代码，写入的第一个字，从%MW8 开始写入的字数。

标准 Modbus 请求

导言

您能使用请求来交换设备间的数据以访问位和字信息。RTU 和 ASCII 模式使用相同的表格式。

格式	参考
位	%Mi, 0x 或 1x 寄存器
字	%MWi, 3x 或 4x 寄存器

Modbus 主模式：读 **N** 位

下表是请求 01 和 02 描述。

	表索引	高字节	低字节
控制表	0	01（发送/接收）	06（发送长度）
	1	00（接收偏移）	00（发送偏移）
发送表	2	从@(1..247)	01 或 02（请求码）
	3	读取的第一位的编号	
	4	N = 读取的位数	
接收表（响应之后）	5	从@(1..247)	01（请求码）
	6	发送的数据字节数（由位组成的一个字节）	
	7	读取的第一个字节 (value = 00 或 01)	读取的第二个字节 (if N>1)
	8	读取的第三个字节 (if N>1)	
	…		
	(N/2)+6	读取的第 N 个字节 (if N>1)	

Modbus 主模式：
读 **N** 字

下表是请求 03 和 04 描述。

	表索引	高字节	低字节
控制表	0	01（发送/接收）	06（发送长度）
	1	03（接收偏移）	00（发送偏移）
发送表	2	从@(1..247)	03 或 04（请求码）
	3	读取的第一字的编号	
	4	N = 读取的字数	
接收表（响应之后）	5	从@(1..247)	03（请求码）
	6	00（由 Rx 偏移加入的字节）	2*N（读取的字节数）
	7	读取的第一个字	
	8	读取的第二个字(if N>1)	
	…		
	N+6	读取的第 N 个字(if N>1)	

注意：Rx 偏移 3 将在接收表的第三个位置加入一个字节（值= 0）。这样保证了读取字节数和读取字的值在表中有一个好的分配。

Modbus 主模式：
写 1 输出位

下表是请求 05 描述。

	表索引	高字节	低字节
控制表	0	01（发送/接收）	06（发送长度）
	1	00（接收偏移）	00（发送偏移）
发送表	2	从@(1..247)	05（请求码）
	3	写的位数	
	4	写的位的值	
接收表（响应之后）	5	从@(1..247)	05（请求码）
	6	被写的位数	
	7	被写的值	

注意：

- 此请求不需要用到偏移。
- 这里响应帧和请求帧一样（正常情况下）。
- 为了写一位 1，发送表中相关字必须包含值 FF00H，为了写位 0 必须包含值 0。

Modbus 主模式：
写 1 输出字

下表是请求 06 描述。

	表索引	高字节	低字节
控制表	0	01（发送/接收）	06（发送长度）
	1	00（接收偏移）	00（发送偏移）
发送表	2	从@(1..247)	06（请求码）
	3	写的字数	
	4	写的字值	
接收表（响应之后）	5	从@(1..247)	06（请求码）
	6	被写的字数	
	7	被写的值	

注意：

- 此请求不需要用到偏移。
- 这里响应帧和请求帧一样（正常情况下）。

Modbus 主模式：写 **N** 位

下表是请求 15 描述。

	表索引	高字节	低字节
控制表	0	01（发送/接收）	8 + 字节数（发送）
	1	00（接收偏移）	07（发送偏移）
发送表	2	从@(1..247)	15（请求码）
	3	写的第一位的编号	
	4	N1 = 写的位数	
	5	00（不发送，偏移结果）	N2 = 写的数据字节数
	6	第一个字节的值	第二个字节的值
控制表	7	第三个字节的值	
	…		
发送表	6+(N2/2)	第 N2 个字节的值	
接收表（响应之后）		从@(1..247)	15（请求码）
		被写的第一位的编号	
		被写的位数（= N1）	

注意：

- Tx 偏移=7 将取消发送帧中的第 7 个字节。这样也是为了发送表中的字值有一个好的对应。

Modbus 主模式：写 **N** 字

下表是请求 16 描述。

	表索引	高字节	低字节
控制表	0	01（发送/接收）	8 + (2*N) （发送长度）
	1	00（接收偏移）	07（发送偏移）
发送表	2	从@(1..247)	16（请求码）
	3	写的第一字的编号	
	4	N = 写的字数	
	5	00（不发送，偏移结果）	2*N = 写的字节数
	6	写的第一个字值	
	7	写的第二个值	
	…		
	N+5	写的第 N 个值	
接收表（响应之后）	N+6	从@(1..247)	16（请求码）
	N+7	被写的第一字的编号	
	N+8	被写的字数（= N）	

注意：Tx 偏移=7 将取消发送帧中的第 5 个 MMSB 字节。这样也是为了发送表中的字值有一个好的对应。

7

概览

本章的主题　　本章描述了怎样管理内置式模拟通道和电位器。

本章包含了哪些内容？　　本章包含了以下主题：

导言

Twido 控制器具有：

- 一个模拟电位器，对于TWDLCAA10DRF，TWDLCAA16DRF控制器，和所有的模块型控制器（TWDLMDA20DTK, TWDLMDA20DUK, TWDLMDA20DRT, TWDLMDA40DTK and TWDLMDA40DUK）。
- 两个电位器，对于TWDLCAA24DRF控制器。

编程

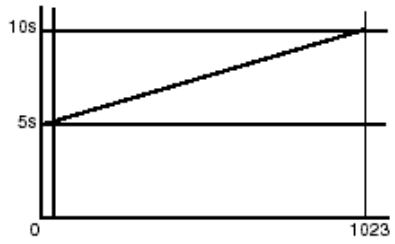
模拟电位器 1 的数值从 0 到 1023，模拟电位器 2 的数值从 0 到 511，对应着电位器提供的模拟值，并包含在下面两个输入字中：

- %IW0.0.0 对应模拟电位器 1（左边）
- %IW0.0.1 对应模拟电位器 2（右边）

这些字可用于算术操作。它们可用于任何形式的调节，例如，预置延迟时间或计数器，调节脉冲发生器的频率或机器预热时间。

示例

用模拟电位器将延迟时间从 5s 调为 10s：

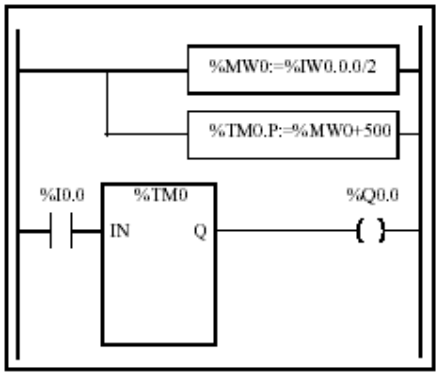
这个调节实际上用到了模拟 电位器 1 的全部调节范围： 从 0 到 1023	

下面参数被选择用于配置时间延迟模块%TM0：

- 类型 TON
- 时基：10 ms

时间延迟的预置值从电位器的调节值计算出来，计算方程为 %TM0.P := (%IW0.0.0/2)+500。

上面示例的代码如下：




```
LD      1
[%MW0:=%IW0.0.0/2]
[%TM0.P:=%MW0+500]
BLK     %TM0
LD      %I0.0
IN
OUT_BLK
LD      Q
ST      %Q0.0
END_BLK
.....
```

模拟通道

导言

所有的模块型控制器(TWDLMDA20DTK, TWDLMDA20DUK, TWDLMDA20DRT,TWDLMDA40DTK, and TWDLMDA40DUK)都有一个内置式模拟通道。其电压输入范围0到10V，数字信号0到511。模拟通道有八路采样，最后采用它们的平均值。

原理

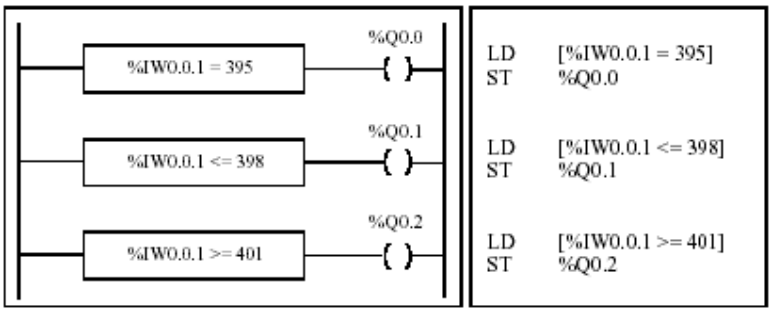
模数转换器将0到10V的输入电压采样为0到511的数字值。该值存储于系统字%IW0.0.1中。由于值的转换是线性的，所以每个增加的数字相当于20 mV（10V/512）。一旦系统检测到值511，通道即视为饱和。

编程示例

控制一个烤炉的温度：烹饪的温度设为 350°C。+/- 2.5°C 的温度变化范围可以得到平稳输出%Q0.0 和%Q0.2。此例中用到了模拟通道的全部范围 0 到 511。各温度点的模拟设置为：

温度（°C）	电压	系统字%IW0.0.1
0	0	0
347.5	7.72	395
350	7.77	398
352.5	7.83	401
450	10	511

上面示例的代码为：



8

概览

本章的主题　　本章提供了 Twido 控制器模拟模块管理概述。

本章包含了哪些内容？　　本章包含了以下主题：

模拟模块概述

导言

除了内置式 10 位电位器和 9 位模拟通道，所有支持扩展 I/O 的 Twido 控制器都能配置模拟 I/O 模块，并和它们通信。

这些模拟模块是：

名字	点数	信号范围	编码
TWDAMI2HT	2 输入	0-10V,4-20 mA	12 位
TWDAM01HT	1 输出	0-10V,4-20 mA	12 位
TWDAMM3HT	2 输入，1 输出	0-10V,4-20 mA	12 位
TWDALM3LT	2 输入，1 输出	0-10V,输入 Th 或 PT100,输出 4-20 mA	12 位

模拟模块操作

输入和输出字(%IW 和 %QW)用于用户程序和模拟模块之间的数据交换。这些字的更新在运行模式下与控制器扫描同步完成。

警告

意外的设备启动

当控制器置于停止模式时，模拟输出将置于可靠位置。在数字输出情形下，可靠位置为 0。

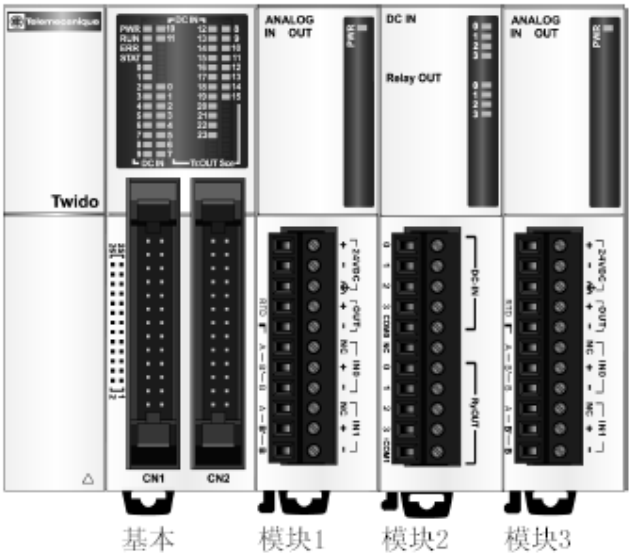
如果不遵守这个警告，将会导致人身伤害或设备损害。

模拟输入和输出寻址

导言　　模拟通道的分配地址取决于它们在扩展总线中的位置。

模拟 I/O 寻址示例

此例中，TWDLMDA40DUK 有一个内置式模拟调节 10 位电位器，一个 9 位内置式模拟通道。扩展总线上配置有：一个 TWDAMM3HT 模拟模块，一个 TWDDMM8DRT 输入/输出继电器模块，和另外一个 TWDAMM3HT 模拟模块。



下表是每个输出寻址的详细情况。

描述	基本	模块 1	模块 2	模块 3
电位器 1	%IW0.0.0			
内置式模拟通道	%IW0.0.1			
通道 1 模拟输入		%IW0.1.0		%IW0.3.0
通道 2 模拟输入		%IW0.1.1		%IW0.3.1
通道 1 模拟输出		%QW0.1.0		%QW0.3.0
通道数字输入			%I0.2.0 - %I0.2.3	
通道数字输出			%Q0.2.0 -%Q0.2.3	

模拟输入和输出配置

导言　　本部分提供了模拟输入和输出的配置信息。

模拟 I/O 配置

模块配置对话框用于管理模拟模块的参数。

注意：当控制器没有连接时，您可以离线修改参数。

模拟通道的分配地址取决于它们在扩展总线中的位置。作为编程帮助，应用程序中可以使用前面定义过的符号来操作数据。

TWDAM01HT, TWDAMM3HT和TWDALM3LT的单输出通道可配置通道类型如下：

- 不被使用
- 0 - 10 V
- 4 – 20 mA

TWDAMI2HT和TWDAMM3HT的双输入通道可配置通道类型如下：

- 不被使用
- 0 - 10 V
- 4 – 20 mA

	注意
	设备损坏 如果您将输入与电压测量值相接，而您的TwidoSoft配置却是电流类型，您将永久性地损害模拟模块。确信连线与TwidoSoft配置一致。 如果不遵守这个警告，将会导致人身伤害或设备损害。

TWDALM3LT 的两输入通道可被配置类型为：

- 不被使用
- 热电偶 K
- 热电偶 J
- 热电偶 T
- PT 100

当通道配置好以后，您能根据下表选择赋值单位和输入范围：

范围	单位	描述
标准	没有	固定范围：最小 0，最大 4095
自定义	没有	由用户定义：不小于-32768，不大于 32767

范围	单位	描述
摄氏温度	0.1°C	国际温度标度。只对 TWDALM3LT 输入通道有用。
华氏温度	0.1°F	此温度标度下水的沸点是 212°F(100°C)，结冰点是 32°F (0°C)。只对 TWDALM3LT 输入通道有用。

模拟模块状态信息

状态信息　　下表是您需要监控的模拟 I/O 模块的状态信息。

系统字	功能	描述
%SW80	基本 I/O 状态	位 [0] 通道处于正常操作（对于通道所有操作） 位 [1] 模块初始化（或所有通道的信息初始化） 位 [2] 硬件故障（外部电源故障，所有通道的普通故障） 位 [3] 模块配置错误 位 [4] 通道0输入数据转换处理中 位 [5] 通道1输入数据转换处理中 位 [6] 通道0输入热电偶没有配置 位 [7] 通道1输入热电偶没有配置 位 [8] 不被使用 位 [9] 没有使用 位 [10] 通道0模拟输入数据超出范围 位 [11] 通道1模拟输入数据超出范围 位 [12] 连线错误（通道0模拟输入数据低于电流范围，电流回路开路） 位 [13] 连线错误（通道1模拟输入数据低于电流范围，电流回路开路） 位 [14] 没有使用 位 [15] 输入通道不可用
%SW81	扩展 I/O 模块 1 状态：定义与%SW80 相同	
%SW82	扩展 I/O 模块 2 状态：定义与%SW80 相同	
%SW83	扩展 I/O 模块 3 状态：定义与%SW80 相同	
%SW84	扩展 I/O 模块 4 状态：定义与%SW80 相同	
%SW85	扩展 I/O 模块 5 状态：定义与%SW80 相同	
%SW86	扩展 I/O 模块 6 状态：定义与%SW80 相同	
%SW87	扩展 I/O 模块 7 状态：定义与%SW80 相同	

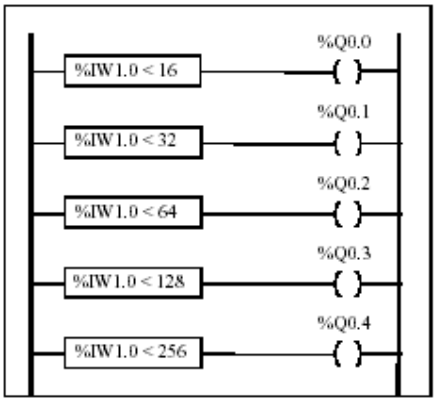
模拟模块使用示例

导言

本部分提供了一个 Twido 可用模拟模块的使用示例。

示例

此例将模拟输入信号与五个阈值进行比较。模拟输入被比较后，如果它小于或等于某个阈值，基本控制器将置上对应位。




```
LD [%IW1.0 < 16]
ST %Q0.0

LD [%IW1.0 < 32]
ST %Q0.1

LD [%IW1.0 < 64]
ST %Q0.2

LD [%IW1.0 < 128]
ST %Q0.3

LD [%IW1.0 < 256]
ST %Q0.4
```

9

概览

本章的主题　　本章提供了 AS-i 主模块 TWDNOI10M3 及其从设备的软件安装信息。

本章包含了哪些内容？　本章包含了以下主题：

导言

AS-i 总线（Actuator Sensor-Interface，执行器传感器接口）允许传感器设备/执行器在自动控制的最底层通过一根电缆互联。
这些传感器/执行器在此文档中定义为从设备。

为了实现 AS-I 应用，您需要定义应用所在的物理环境（扩展总线，电源，处理器，模块，连接到总线的 AS-I 从设备），然后保证软件的可执行性。

第二个方面即软件可通过不同的TwidoSoft编辑器来实现：

- 或者本地模式，
- 或者在线模式。

AS-i V2 总线

AS 接口主模块 **TWDNOI10M3** 包含下列功能：

- M3 表：此表包含 AS-i V2 标准定义的所有功能，但不支持 S7-4 模拟表。
- 每个模块有一个 AS-i 通道
- 自动寻址地址为 0 的从设备
- 管理表和参数
- 总线输入的极性反向保护

AS-i 总线允许：

- 最多 31 个标准地址从设备和 62 个扩展地址从设备
- 最多 248 输入和 186 输出
- 最多 7 个从设备模拟量（每个从设备最多 4 I/O）
- 最大循环时间 10 ms

一个 Twido 模块型控制器或一个 LCAA24DRF 一体型控制器最多可连接 2 个 AS-i 主模块。

总的功能描述

总的介绍

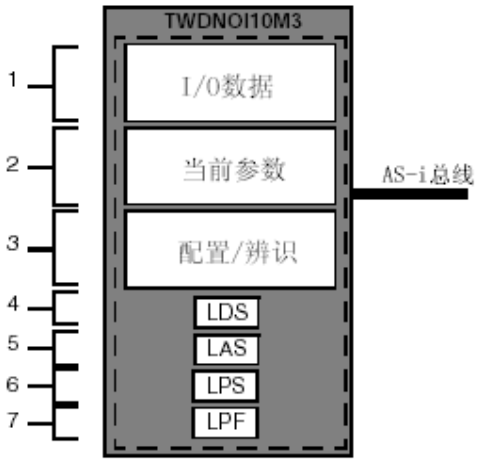
对于 AS-I 配置，TwidoSoft 软件允许用户：

- 手动配置总线（从设备声明和总线中地址分配）
- 根据总线当前情况修改配置
- 控制总线状态

基于这个原因，所有来或去 AS-i 主模块的数据将存储在特定对象（字和位）中。

AS-i 主机结构

AS-i 模块包含数据区，允许您管理从设备列表和输入/输出数据映像。这些信息存储在易失存储器中。下图显示了 **TWDNOI10M3** 模块结构。



注释：

地址	条目	描述
1	I/O 数据 (IDI, ODI)	AS-i V2 总线 248 输入和 186 输出映像。
2	当前参数 (PI, PP)	所有从设备参数映像。
3	配置/辨识 (CDI, PCD)	此区域包含所有检测到的从设备的 I/O 代码和辨识代码。
4	LDS	总线上所有检测到的从设备列表。
5	LAS	总线上所有活动从设备列表。
6	LPS	总线上提供且通过 TwidoSoft 配置的从设备列表。
7	LPF	具有设备故障的从设备列表。

从设备结构

标准地址从设备均具有：

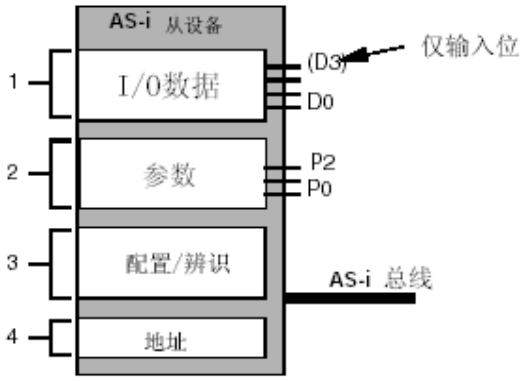
- 4个输入/输出位
- 4个参量位

扩展地址的从设备均具有：

- 4个输入/输出位（最后一位保留，仅供登陆使用）
- 3个参量位

每个从设备都有自己的地址，表和子表（定义变量交换）。

下图显示了一个扩展地址从设备的结构：



注释：

地址	条目	描述
1	输入/输出数据	输入数据被从设备存储并可为 AS-i 主模块可用。 输出数据由主模块更新。
2	参数	参数用于控制和转换内部工作模式到传感器或执行器。
3	配置/辨识	此区域包含： - 对应 I/O 配置的代码， - 从设备辨识（ID）代码， - 从设备辨识代码（ID1 和 ID2）。
4	地址	从设备的物理地址。

注意：工作参数，地址，配置和辨识数据都存储在永久性存储器中。

软件安装原则

概览

根据 TwidoSoft 哲学，用户应该采用渐进的方法来创建 AS-I 应用程序。

安装原则

用户必须知道怎样配置他的 AS-i 总线（见怎样在现有 AS-i V2 配置中插入从设备）。
下表显示了 AS-i 总线软件执行的不同阶段。

模式	阶段	描述
本地	模块声明	AS-i 主模块 TWDNOI10M3 在扩展总线上的位置选择。
	模块通道配置	“主”模式选择。
	从设备声明	为每个设备选择： I 总线上的位置编号， I 从设备标准或扩展地址类型。
	配置参数确认	从设备级确认。
	应用程序 全局确认	应用程序级确认。
本地或被连接	符号化（可选）	从设备相关变量符号化。
	编程	AS-i V2 功能编程。
被连接	传递	传递应用程序到 PLC。
	调试	通过下面帮助调试应用程序： I 调试屏幕，一方面用于显示从设备（地址，参数），另一方面用于分配它们相应的地址， I 诊断屏幕允许辨识错误。

注意：扩展总线上 AS-i 主模块的声明和删除与其它扩展模块一样。但是，一旦扩展总线上声明了两个 AS-i 主模块，TwidoSoft 将不允许再声明其它 AS-i 主模块。

连接之前注意　　在连接（通过软件）PC 到控制器之前，并为了避免出现问题：

- 确信总线上不物理存在地址为 0 的从设备。
- 确信不物理存在 2 个地址相同的从设备。

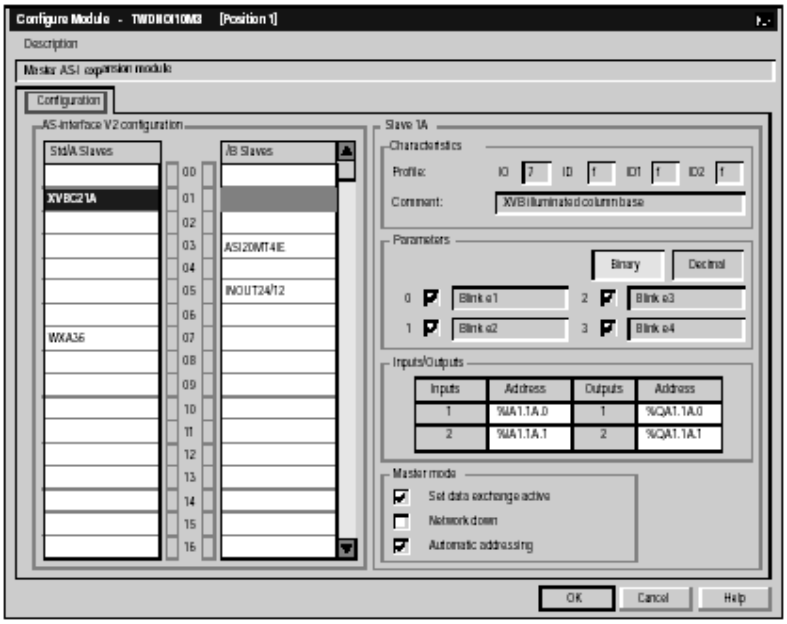
AS-i 总线配置屏幕描述

概览

AS-i 主模块配置屏幕可以访问模块和从设备相关参数。
它可以在本地模式下显示和修改参数。

本地模式图例

本地模式下配置屏幕图例：



本地模式下屏幕描述

该屏幕将总线所有数据组成三块信息：

块	描述
AS-接口 V2 配置	用户期望的总线映像：总线上从设备的标准和扩展地址设置观察。向下移动垂直条光标到达下面地址。变成灰色的地址对应从设备配置中不可用的地址。例如，如果一个新的从设备标准地址设置被声明为 1A，则地址 1B 自动变成灰色。
从设备 xxA/B	被选从设备配置： I 特性：IO 码，ID 码，ID1 和 ID2 码（表），以及从设备注释 I 参数：参数列表（可修改），根据用户判断用二进制（4 个检查框）或十进制（1 个检查框）表 I 输入/输出：可用的输入/输出列表及它们的地址
主模式	AS-i 模块的三种可用功能是可以激活或不激活的（例如，自动寻址）。 “设置数据交换激活”和“自动寻址”模式默认被选中。

屏幕还包含 3 个按钮：

按钮	描述
OK	用于保存配置屏幕上显示的 AS-i 总线配置。 然后返回到主屏幕。 配置可被传送到Twido控制器。
Cancel	不承认修改并返回到主屏幕。
Help	打开屏幕帮助窗口。

注意：配置屏幕中的修改只能在本地模式下进行。

AS-i 总线配置

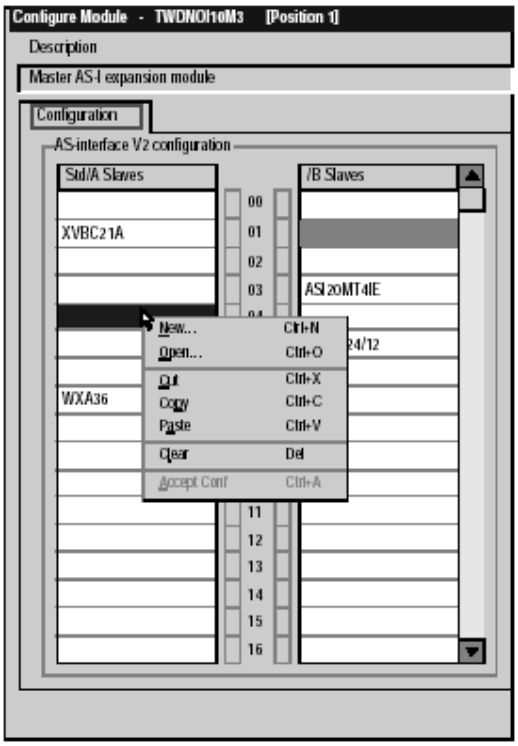
导言

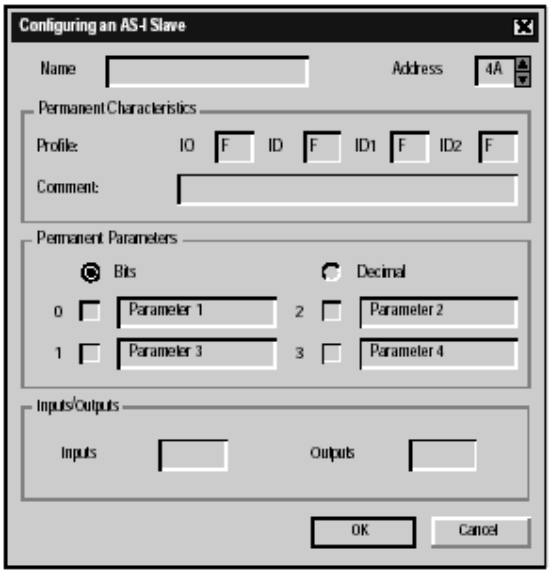
AS-i 总线配置在本地模式下通过配置屏幕完成。

一旦 AS-i 主模块和主模块模式被选定，AS-i 总线配置将由从设备配置组成。

从设备声明和配置过程

AS-i V2 总线从设备创建或修改过程：

步骤	动作
1	在总线映像的期望地址单元（没有变成灰色）： ❙ 双击：到达步骤 3 或 ❙ 右击： 结果：  注释： 出现一个快捷菜单。用于： ❙ 配置总线新的从设备 ❙ 修改相应从设备的配置 ❙ 复制（或 Ctrl+C），剪切（或 Ctrl+X），粘贴一个从设备（或 Ctrl+V） ❙ 删除一个从设备（或 Del）

步骤	动作
2	在快捷菜单中选择： l "New"创建一个新的从设备：一个从设备配置屏幕显示出来；"Address"区域显示被选地址，"Profile"区域被默认设置为 F，屏幕其它区域均为空白。 l "Open"创建一个新的从设备或修改被选从设备的配置。对于新的从设备，一个新的从设备配置屏幕显示出来，"Address"区域显示被选地址，"Profile"区域被默认设置为 F，屏幕其它区域均为空白。对于修改，从设备配置屏幕区域显示被选从设备以前的定义值。 新的从设备配置屏幕图例： 
3	在显示的从设备配置屏幕中，输入或修改： l 新表的名字（13 字符以内）， l 注释（可选）。
4	输入： l IO 代码（对应输入/输出配置）， l ID 代码（标识符），（加上 ID1 表示扩展类型）。 注释： "Inputs"和"Outputs"区域显示输入和输出通道数。当 IO 代码输入后，它们将被自动执行。

步骤	动作
5	对每个参数定义： l 系统确认（"Bits"视图被选框，或"Decimal"视图中 0 到 15 之间的十进制值）， l 比"Parameter X"更有明确意义的名字（可选）。 注释： 被选参数是提供给 AS-i 主模块的永久参数的映像。
6	如果需要，可以通过点击地址左边的增加/减少箭头（给定被授权的地址）或用键盘输入地址，来修改"Address"（在总线可用地址限制之内）。
7	点击"OK"按钮确认从设备配置。 结果将确认： l IO 和 ID 被授权， l 根据 ID 代码（如果 ID 代码等于 A，只有"bank" /B 从设备可用）授权从地址（如果使用键盘）。 如果出错，将有一个错误消息警告用户（例如：“该从设备不能使用这个地址”），且屏幕重新显示初始化值（根据错误出现表值或地址值）。

注意：软件限定模拟从设备声明数目不超过 7。

调试屏幕描述

概览

当 PC 与控制器相连接（在上载应用程序到控制器之后），"Debug"标签出现在"Configuration"的右边；它允许进入调试屏幕。

调试屏幕动态地提供物理总线的映像，包括：

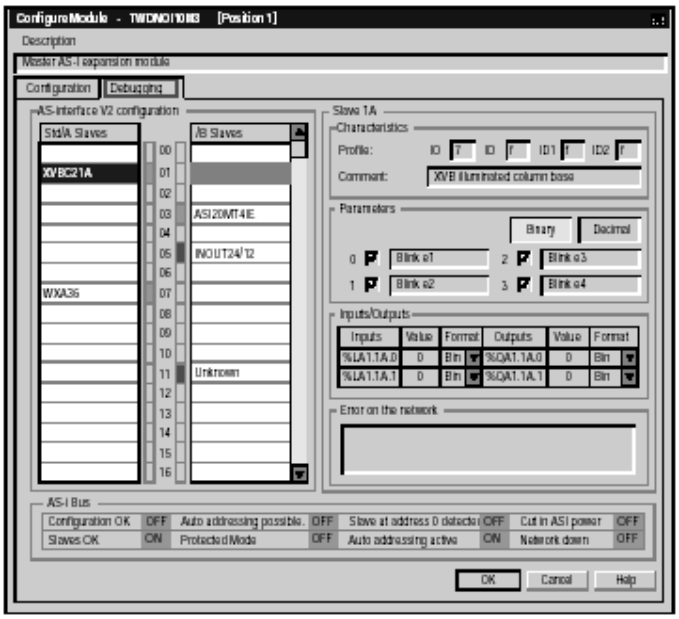
- 配置中指定从设备（被输入）列表及它们的名字，和被检测的从设备（名字未知，但其它均已指定）列表
- AS-i 模块和从设备状态
- 被选从设备的表，参数和输入/输出值的映像。

它可以让用户：

- 获得出错从设备的诊断（见从设备状态显示，p.168），
- 连接模式下修改从设备地址（见从设备地址修改，p.169），
- 传递从设备的映像到配置屏幕（见在线模式下AS-i总线配置更新，p.172），
- 用指定地址寻址所有从设备（第一次调试时）。

“调试”屏幕图例

调试屏幕（仅连接模式下）图例如下：



调试屏幕描述

“调试”屏幕提供了配置屏幕相同的信息（见本地模式下屏幕描述，p.161）。

下表列出了不同之处：

目录	描述
AS-接口 V2 配置	物理总线映像。 包括从设备状态： ❙ 绿色指示灯：此地址从设备处于活动状态。 ❙ 红色指示灯：此地址从设备出错，且消息通知您错误类型在"Error on the network"中。
从设备 xxA/B	被选从设备配置映像： ❙ 特性：检测到的表映像（变成灰色，不能修改） ❙ 参数：检测到的参数映像。用户只能选择参数显示格式 ❙ 输入/输出：检测到的输入/输出值显示，不能修改
网络错误	通知您错误类型，如果被选从设备出错。
AS-i 总线	"Read Status"命令结果通知。 ❙ 显示总线状态：例如，"Configuration OK = OFF"表示用户指定配置与总线物理配置不对应。 ❙ 显示AS-i主模块被授权的功能：例如，"Automatic addressing active = ON"表示主模块自动寻址被授权。

从设备状态显示

当对应地址的指示灯变红时，说明该地址的从设备出错。"Error on the network"窗口提供了被选从设备的诊断。

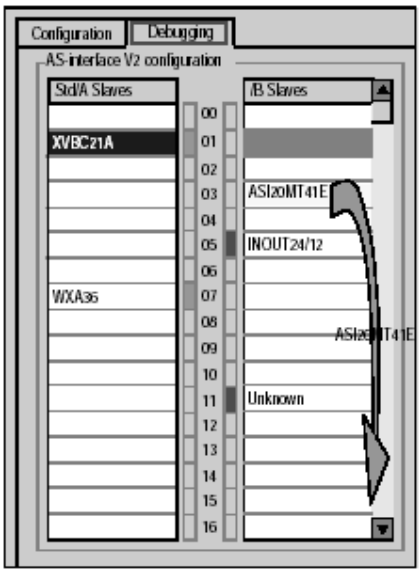
错误描述：

- ❙ 用户对给定地址的配置中的指定表与总线中该地址被检测到的实际表不一致（诊断："Profile error"）
- ❙ 总线检测到一个新的设备，没有在配置中指定：红色指示灯显示该地址且从设备名字显示“未知”（诊断："Slave not projected"）
- ❙ 外围设备故障，如果从设备检测到（诊断："Peripheral fault"）
- ❙ 配置表被指定，但总线没有检测到该地址的从设备（诊断："Slave not detected"）。

从设备地址修改

概览　　从调试屏幕，用户可以在在线模式下修改从设备的地址。

从设备地址修改　下表显示了从设备地址修改过程：

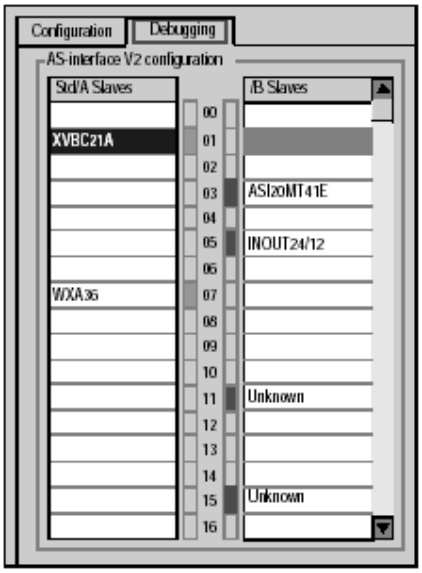
步骤	描述
1	进入"Debug"屏幕。
2	在"AS-interface V2 Configuration"选择从设备。
3	拖放从设备到期望地址对应的单元。 图例：拖放从设备 3B 到 15B 

步骤	描述

结果：

所有的从设备参数被自动检查以示该操作是否可行。

结果图例：



该操作完成后，地址3B的从设备诊断显示"slave not detected"表示该地址对应的从设备不再存在。选择地址15B，被移动的从设备的表和参数被重新部署，但是从设备的名字为未知，因为该地址名字没有被指定。

注意：一个从设备的表和参数不与其名字关联在一起。不同名字的几个从设备可以具有相同的表和参数。

在线模式下 AS-i 总线配置更新

概览

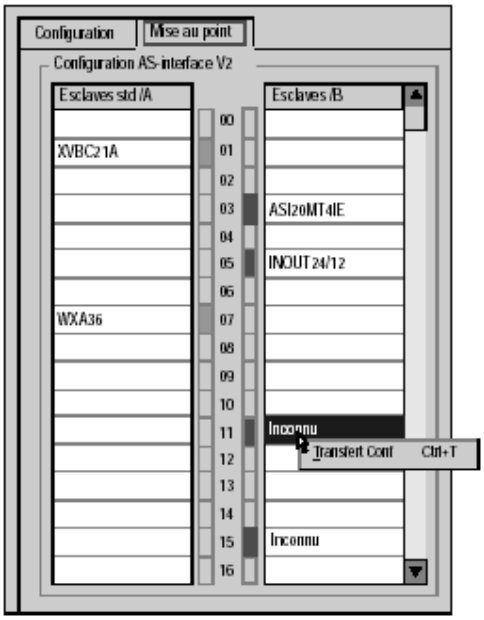
在线模式下，配置屏幕的修改不被授权且物理配置和软件配置可以不同。配置屏幕可以考虑已配置或未配置从设备的表或参数的不同；实际上，在传递新的应用程序到控制器之前，可以发送修改到配置屏幕。下面是跟踪考虑物理配置的过程：

步骤	描述
1	传递期望从设备配置到配置屏幕。
2	配置屏幕中配置接受。
3	新配置确认。
4	传递应用程序到模块。

传递从设备映像到配置屏幕。

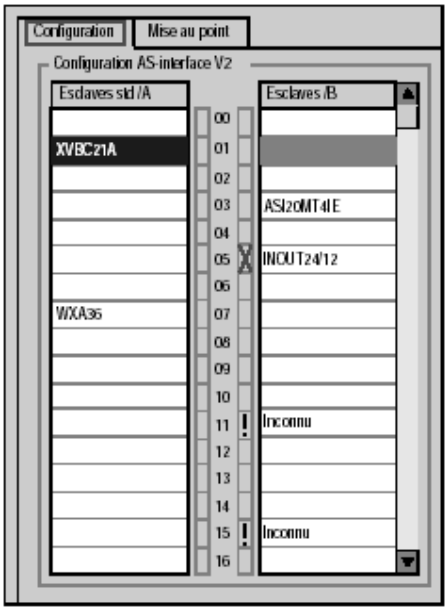
当配置中未指定的从设备被总线检测到时，调试屏幕的"AS-interface V2 Configuration zone"中该检测地址将出现一个“未知”从设备。

下表描述了传递“未知”从设备映像到配置屏幕的过程：

步骤	描述
1	进入"Debug"屏幕。
2	在"AS-interface V2 Configuration"区域选择期望的从设备。
3	右击鼠标选择"Transfer Conf"。 图例：  结果： 被选从设备的映像（表和参数的映像）被传递到配置屏幕。
4	对每个您想传递其映像到配置屏幕的从设备重复上述操作。

返回到配置屏幕

当用户返回到配置屏幕，所有被传递的新的从设备（没被指定）均可见。传递所有从设备后的配置屏幕图例：



注释：

- 叉号表示被传递的从设备的表映像和配置屏幕开始期望的表有差别。
- 感叹号表示配置屏幕中加入了新表。

解释：

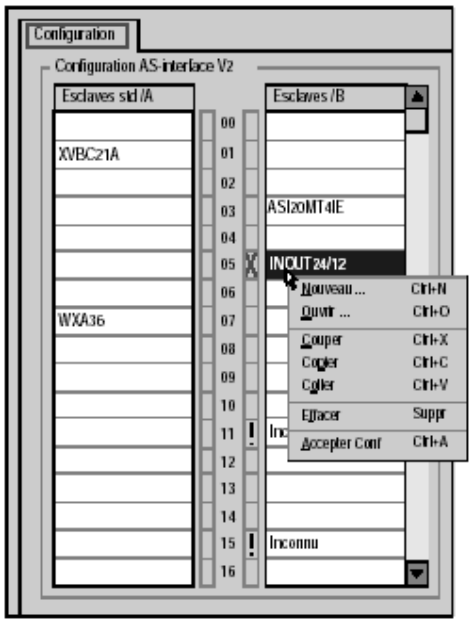
配置屏幕一般显示期望配置的永久映像（这是尽管地址改变（见从设备地址修改，p.170）但从设备依然存在于3B的原因），直到被总线的当前映像显示为止。

期望的从设备的表和参数显示与期望值一致。未知从设备的表和参数显示与被检测到的映像一致。

传递最终应用程序到模块的过程

在传递新的应用程序到模块之前，用户对于每个从设备可以接受检测到的表和参数（被传递到配置屏幕），或手动修改配置（见从设备声明和配置过程，p.163）。

下表描述了确认和传递最终配置到模块的步骤：

步骤	动作
1	通过软件，断开PC和模块连接。 注意： PC与模块相连时配置屏幕不能实现修改。
2	右击期望的从设备。
3	2个选择： I 选择"Accept Conf"接受被选从设备被检测到的表。 图例：  对每个标记叉号的从设备，将有一个消息警告用户此操作将从设备的初始表（显示在屏幕上）。 I 在右击菜单上选择其它选项手动配置被选从设备。

步骤	动作
4	对每个配置中的期望从设备重复上述操作。
5	点击"OK"按钮确认和创建新的应用程序。 结果：自动返回到主屏幕。
6	传递应用程序到模块。

AS-i V2 从设备自动寻址

概览

每个 AS-i 总线从设备必须分配（通过配置）一个唯一的物理地址。它必须与 TwidoSoft 声明的一样。

TwidoSoft 软件提供了从设备自动寻址功能，因此没有必要使用 AS-i 控制台。

自动寻址功能用于：

- 更换故障从设备，
- 插入新的从设备。

过程

下表显示了自动寻址参数设置过程。

步骤	动作
1	进入 AS-i V2 主模块配置屏幕。
2	点击主模式区域的自动寻址检查框。 结果：自动寻址功能被激活（框被选中）或停止（框没被选中）。 注意：默认方式下，自动寻址参数在配置屏幕中已被选择。

怎样在现有 AS-i V2 配置中插入从设备

概览

可以不用袖珍编程器而插入一个设备到 AS-i V2 配置中。

此操作可以进行，一旦：

- 配置模式中自动寻址功能被激活（见 AS-i V2 从设备自动寻址，p.177），
- 物理配置中不存在单一的从设备，
- 将要插入的从设备已在屏幕配置中指定，
- 从设备具有配置期望的表，
- 从设备地址为 0 (A)。

AS-i V2 模块将自动赋给从设备配置预置值。

过程

下表显示了自动插入一个新的有效从设备的过程。

步骤	动作
1	本地模式下在配置屏幕中加入新的从设备。
2	连接模式下完成配置传递给 PLC。
	物理连接地址为 0 的新的从设备到 AS-i V2 总线。

注意：如果必要，应用程序可被上面操作执行修改多次。

故障 AS-i V2 从设备的自动替换

原理

当一个从设备被声明出错，它能被一个相同类型的从设备自动替代。
这个过程不需要 AS-i V2 总线停止，也不需要任何操作，如果配置模式的自动寻址功能被激活（见 AS-i V2 从设备自动寻址，p.177）。

两个选择可用：

- 替代从设备使用袖珍编程器编程，与故障从设备地址相同，表和子表也相同。这样自动插入到被检测从设备列表（LDS）和活动从设备列表（LAS），
- 替代从设备空白（地址 0 (A)，新的从设备）且和故障从设备表相同。替换从设备的地址将自动设定，然后插入到被检测从设备列表（LDS）和活动从设备列表（LAS）。

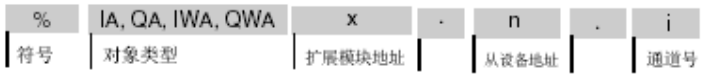
连接 AS-i V2 总线的从设备的相关 I/O 寻址

概览

这页介绍了有关从设备数字或模拟 I/O 寻址的详细资料。
为了避免远程 I/O 的混乱，可用新的符号，例如 AS-i 语法：%IA。

图例

寻址原理表示：



详细值

下表给出了 AS-i V2 从设备对象的详细值：

条目	值	注释
IA	-	从设备物理数字输入映像。
QA	-	从设备物理数字输出映像。
IWA	-	从设备物理模拟输入映像。
QWA	-	从设备物理模拟输出映像。
x	1 到 7	扩展总线 AS-I 模块地址。
n	0A 到 31B	位置 0 不能被配置。
i	0 到 3	-

示例

下表是一些 I/O 寻址示例。

I/O 对象	描述
%IWA4.1A.0	扩展总线上位置为 4 的 AS-i 模块的地址 1A 的从设备的 0 号模拟输入。
%QA2.5B.1	扩展总线上位置为 2 的 AS-i 模块的地址 5B 的从设备的 1 号数字输出。
%IA1.12A.2	扩展总线上位置为 1 的 AS-i 模块的地址 12A 的从设备的 2 号数字输入。

后台交换

下面描述的对象在后台交换，换句话说，它们在每个 PLC 循环被自动交换。

直接交换

AS-i 总线的相关对象（字和位）为实现 AS-I 功能的高级编程贡献数据（例如：总线操作，从设备状态，等等）和其它命令。

这些对象通过扩展总线在Twido控制器和AS-i主模块之间直接交换：

- 程序用户用指令的方式请求：ASI_CMD（见下面指令“ASI_CMD介绍”）
- 通过调试屏幕或活动表。

保留的特殊系统字

Twido 控制器为主模块保留的系统字使您决定网络的状态：%SW73 保留给第一个 AS-i 扩展模块，%SW74 保留给第二个。这些字只有前 5 位可用；它们只读。

下表显示了可使用的位：

系统字	位	描述
%SW73 和 %SW74	0	系统状态（=1 如果配置正确，否则为 0）
	1	数据交换（=1 可以数据交换，0 如果处于数据交换关闭模式（见 AS-i V2 总线接口模块操作模式，p.187））
	2	系统被停止（=1 如果离线（见离线模式，p.187）模式可用，否则 0）
	3	ASI_CMD 指令被终止（=1 如果被终止，0 如果进行中）
	4	ASI_CMD 错误指令（=1 如果指令出错，否则 0）

使用示例（对第一个 AS-i 扩展模块）：

在使用 ASI_CMD 指令之前，必须检查位%SW73:X3 以示指令是否不处于进行状态：确认%SW73:X3 = 1。

为了确定指令是否已被正确执行，确认位%SW73:X4 等于 0。

ASI_CMD 指令介绍

对每个用户程序，ASI_CMD指令允许用户对其网络编程并获得从设备诊断。指令参数通过内部字（存储字）**%MW**x传递。

指令语法如下：

ASI_CMDn **%MW**x:l

标注：

符号	描述
n	AS-i 扩展模块地址（1 到 7）。
x	参数传递地第一个内部字（存储字）的编号（0 到 254）。
l	字数长度指令（2）。

ASI_CMD 指令使用

下表描述了必要时对应参数%MW(x)和%MW(x+1)值的ASI_CMD指令动作。对从设备诊断请求，结果返回通过%MW(x+1)返回。

%MWx	%MWx+1	动作
1	0	退出离线模式。
1	1	转换到离线模式。
2	0	禁止主模块及其从设备间数据交换（进入数据交换关闭模式）。
2	1	允许主模块及其从设备间数据交换（退出数据交换关闭模式）。
3	保留	-
4	结果	读活动从设备列表（LAS表），地址从0A到15A（每个从设备1位）。
5	结果	读活动从设备列表（LAS表），地址从16A到31A（每个从设备1位）。
6	结果	读活动从设备列表（LAS表），地址从0B到15B（每个从设备1位）。
7	结果	读活动从设备列表（LAS表），地址从16B到31B（每个从设备1位）。
8	结果	读被检测从设备列表（LDS表），地址从0A到15A（每个从设备1位）。
9	结果	读被检测从设备列表（LDS表），地址从16A到31A（每个从设备1位）。
10	结果	读被检测从设备列表（LDS表），地址从0B到15B（每个从设备1位）。
11	结果	读被检测从设备列表（LDS表），地址从16B到31B（每个从设备1位）。
12	结果	读从设备外围故障列表（LPF表），地址从0A到15A（每个从设备1位）。
13	结果	读从设备外围故障列表（LPF表），地址从16A到31A（每个从设备1位）。
14	结果	读从设备外围故障列表（LPF表），地址从0B到15B（每个从设备1位）。
15	结果	读从设备外围故障列表（LPF表），地址从16B到31B（每个从设备1位）。
16	结果	读总线状态。 见下节详细结果。

注意：总线状态在每个 PLC 扫描时被更新。但是 ASI_CMD 总线读取指令的结果仅在下一个 PLC 扫描结束时才可得到。

ASI_CMD 指令读取总线状态的详细结果

ASI_CMD 指令读取总线状态（参数%MWx 值等于 16）时，字%MWx+1 的结果格式如下：

%MWx+1		表示（1=是，0=不是）
低字节	第 0 位	配置正确
	第 1 位	LDS.0（地址 0 从设备存在）
	第 2 位	自动寻址激活
	第 3 位	自动寻址可用
	第 4 位	配置激活
	第 5 位	正常操作激活
	第 6 位	APF（电源供应问题）
	第 7 位	离线准备
高字节	第 0 位	外围故障
	第 1 位	数据交换激活
	第 2 位	离线模式
	第 3 位	离线模式（0）/正常模式（1）
	第 4 位	与 AS-i 主模块通信故障
	第 5 位	ASI_CMD 指令进行中
	第 6 位	ASI_CMD 指令错误

ASI_CMD 指令读取总线状态的详细结果

ASI_CMD 指令诊断从设备（%MWx 值在 4 和 15 之间）时，从设备状态通过字%MWx+1 的位（1=正常）来返回。下表给出了对应字%MWx 值的详细结果：

%MWx	%MWx+1															
值	高有效字节								低有效字节							
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
4, 8, 12	15A	14A	13A	12A	11A	10A	9A	8A	7A	6A	5A	4A	3A	2A	1A	0A
5, 9, 13	31A	30A	29A	28A	27A	26A	25A	24A	23A	22A	21A	20A	19A	18A	17A	16A
6, 10, 14	15B	14B	13B	12B	11B	10B	9B	8B	7B	6B	5B	4B	3B	2B	1B	0B
7, 11, 15	31A	30B	29B	28B	27B	26B	25B	24B	23B	22B	21B	20B	19B	18B	17B	16B

为了读取从设备 20B 是否处于活动状态，必须执行 ASI_CMD 指令，内部字%MWx 值为 7。结果通过内部字%MWx+1 返回；从设备 20B 的状态由低字节的第 4 位的值给定：如果第 4 位等于 1，从设备 20B 处于活动状态。

ASI_CMD 指令编程示例

强制 AS-i 主模块（扩展总线上位置为 1）变换到离线模式：

```
LD 1
[%MW0 := 16#0001 ]
[%MW1 := 16#0001 ]
LD %SW73:X3 //如果没有ASI_CMD 指令在进行中，那么继续
[ASI_CMD1 %MW0:2] //强制转换到离线模式
```

读取从设备活动表，地址0A到15A：

```
LD 1
[%MW0 := 16#0004 ]
[%MW1 := 16#0000 //optional]
LD %SW73:X3 //如果没有ASI_CMD 指令在进行中，那么继续
[ASI_CMD1 %MW0:2] //读 LAS 表，地址 0A 到 15A
```

AS-i V2 总线接口模块工作模式

概览

AS-i 总线接口模块 TWDNOI10M3 有三种工作模式，每种对应着特殊的需求。这些模式是：

- 保护模式，
- 离线模式，
- 数据交换关闭模式。

用户程序中可使用 ASI_CMD（见 ASI_CMD 指令介绍，p.183）指令可以进入或退出这些模式。

保护模式

受保护的工作模式是应用程序运行的一般使用模式。假定 AS-i V2 模块在 TwidoSoft 中配置。它：

- 不断地检查被检测从设备列表与期望从设备列表相同，
- 监控电源供应。

这个模式下，从设备只有在配置中被声明和被检测到后才能被激活。

上电或配置阶段，Twido 控制器 AS-i 强制模块进入保护模式。

离线模式

当模块进入离线模式，它首先将所有存在的从设备复置到零且停止总线上的交换。离线模式下，输出被强制到零。

除了使用 TWDNOI10M3 AS-I 模块的 PB2 按钮，还可以通过软件使用 ASI_CMD（见 ASI_CMD 指令编程示例，p.186）指令进入离线模式，这种方法也可以退出该模式并返回到保护模式。但是无论您使用哪种方法（按按钮或 ASI_CMD），如果主模块“数据交换设置激活”在 AS-i 模块配置屏幕中被激活，才能进入该模式。

数据交换关闭模式

当数据交换关闭模式运行时，总线交换继续进行，但数据不再被更新。只能使用 ASI_CMD（见 ASI_CMD 指令使用，p.184）指令进入该模式。

10

概览

本章的主题　　本章提供了使用可选 Twido 操作显示的详细资料。

本章包含了哪些内容？　　本章包含了以下主题：

操作显示

导言

操作显示是一个 Twido 选件，用于显示和控制应用程序数据和一些控制器功能如运行状态和实时钟（RTC）。该选件可以是一体型控制器的卡（TWDXCPODC）或者模块型控制器的扩展模块（TWDXCPODM）。

操作显示有两种工作模式：

- 显示模式：仅显示数据。
- 编辑模式：允许改变数据。

> 注意：操作显示在控制器扫描循环的特定时间间隔被更新。这将在解释专用输出%PLS 或%PWM 输出的显示时导致混乱。这些输出被采样时，它们的输出将总为零，并且显示这个值。

显示及功能

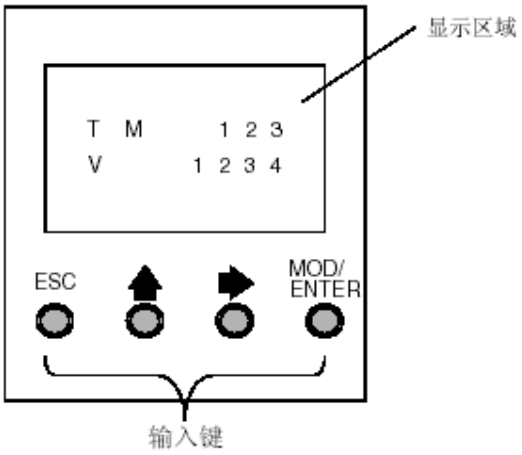
操作显示提供了下面各个显示，以及完成每个显示相应的功能。

- 控制器标识和状态信息：操作显示
  显示固件修订本及控制器状态。用运行，初始化，停止命令改变控制器状态。
- 系统对象和变量：数据显示
  通过地址%I, %Q 和基本控制器的其它软件对象选择应用程序数据。监控和改变所选软件数据对象的值。
- 串口设置：通信显示
  显示和修改通信串口设置。
- 日期时钟定时：时间/日期显示
  显示和配置当前日期和时间（如果 RTC 已安装）。
- 实时修正：RTC 因素
  显示和修改选件 RTC 的 RTC 修正值。

> 注意：日期时钟定时和实时修正只有在实时时钟（RTC）选件卡(TWDXCPRTC)已安装时才可用。

图例

下面图例显示了操作显示视图，由一个显示区域和四个可按按钮输入键组成。



显示区域

该操作显示提供了一个可以显示两行字符的 LCD 显示器：

- 显示器的第一行有三个 13 段字符和四个 7 段字符。
- 第二行有一个 13 段字符，一个 3 段字符（表示正/负号），和五个 7 段字符。

输入键

四个输入可按按钮的功能取决于操作显示模式。

键	显示模式下	编辑模式下
ESC		放弃修改并返回到以前显示。
▲		进入被编辑对象的下一值。
▶	前进到下一显示。	进入编辑的下一对象类型。
MOD/ ENTER	进入编辑模式。	接受修改并返回到以前显示。

选择及操纵显示

操作显示的初始显示器或屏幕显示控制器标识和状态信息。按下▶按钮顺序进入每个显示。如果控制器没有检测到选件 RTC 卡(TWDXCPRTC)，将不会显示日期时钟时间或实时修正因素屏幕。

一个捷径是按ESC键返回到初始显示屏幕。对大部分屏幕，按下ESC键将返回到控制器标识和状态信息屏幕。只有当系统对象和变量编辑没有初始输入（%I0.0.0）时，按下ESC将让您进行第一次或初始系统对象输入。若是修改对象值，不是按可按▶按钮进入第一次值数字，而是重新按MOD/ENTER键。

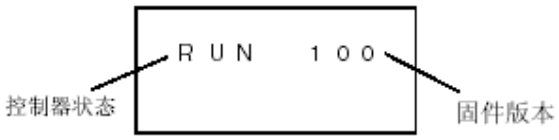
控制器标识和状态信息

导言

Twido 可选操作显示的初始显示器或屏幕显示控制器标识和状态信息。

示例

固件修订本被显示在显示区域的右上角，并且控制器状态显示在显示区域的左上角，如下所示：



控制器状态

控制器状态包含如下：

- NCF：没被配置

控制器处于 NCF 状态直到应用程序被装载。应用程序装载之前不允许其它状态。您可以通过修改系统位 S8（见系统位(%S)，p.454）来测试 I/O。

- STP：被停止

一旦应用程序存在于控制器中，状态改变到 STP 或被停止状态。在此状态，应用程序不运行。输入被更新且数据值保存它们最近的值。输出在此状态不被更新。

- INI：初始

您能只能从STP状态选择修改控制器到INI或初始状态。应用程序不运行。控制器的输入不被更新且数据值设为它们的初始状态。此状态输出不被更新。

- RUN：运行

当处于RUN或运行状态时应用程序运行。控制器的输入被更新且数据值根据应用程序被设置。这是输出被更新的唯一状态。

- HLT：暂停（用户程序错误）

如果控制器进入ERR或出错状态，应用程序被暂停。输入被更新且数据值保存它们最近的值。此状态输出不被更新。这个模式下，错误代码以一个无符号十进制值显示在操作显示的右下部分。

- NEX：不可执行（不可执行）

根据用户逻辑进行在线修改。结果：应用程序不再被执行。它将不能返回状态直到引起不可执行状态的所有原因被解决。

显示和改变控制器状态

使用操作显示，您能改变 STP 状态到 INI 状态，或 STP 到 RUN，或 RUN 到 STP。改变控制器状态如下。

步骤	动作
1	按下➡键直到操作显示可见（或按 ESC）。当前控制器状态显示在显示区域的左上角。
2	按下 MOD/ENTER 键进入编辑模式。
3	按⬆键选择一个控制器状态。
4	按 MOD/ENTER 键接受修改值，或按 ESC 键编辑模式中的任何修改。

导言

可选操作显示为监控和调节应用程序数据提供了这些特性：

- 通过地址（如%I 或 %Q）选择应用程序数据。
- 监控所选软件对象/变量值。
- 修改当前显示的数据对象值（包括强制输入和输出）。

系统对象和变量　系统对象和变量可以被操作显示显示和修改，按照被访问顺序，列表如下。

对象	变量/属性	描述	访问
输入	%I.x.y.z	值	读/强制
输出	%Q.x.y.z	值	读/写/强制
定时器	%TMX.V	当前值	读/写
	%TMX.P	预置值	读/写
	%TMX.Q	完成	读
计数器	%Cx.V	当前值	读/写
	%Cx.P	预置值	读/写
	%Cx.D	完成	读
	%Cx.E	空	读
	%Cx.F	满	读
存储位	%Mx	值	读/写
存储字	%MWx	值	读/写
常量字	%KWx	值	读
存储双字	%MDx	值	读/写
常量双字	%KDx	值	读
存储浮点字	%MFx	值	读/写
常量浮点字	%KFx	值	读
系统位	%Sx	值	读/写
系统字	%SWx	值	读/写
模拟输入	%IW.x.y.z	值	读
模拟输出	%QW.x.y.z	值	读/写
高速计数器	%FCx.V	当前值	读
	%FCx.P	预置值	读/写
	%FCx.D	完成	读
超高速计数器	%VFCx.V	当前值	读/写
	%VFCx.P	预置值	读/写
	%VFCx.U	计数方向	读
	%VFCx.C	捕获值	读
	%VFCx.S0	阈值值 0	读/写
	%VFCx.S1	阈值值 1	读/写
	%VFCx.F	溢出	读
	%VFCx.M	次数完成	读/写
	%VFC.T	时基	读/写
	%VFC.R	输出映像使能	读/写
	%VFC.S	输入映像使能	读/写

对象	变量/属性	描述	访问
输入网络字	%INWx.z	值	读/写
输出网络字	%QNWx.z	值	读/写
Grafcet	%Xx	步位	读
脉冲发生器	%PLS.N %PLS.P %PLS.D %PLS.Q	脉冲数 预置值 完成 当前输出	读/写 读/写 读 读
脉宽调节器	%PWM.R %PWM.P	比率 预置值	读/写 读/写
磁鼓控制器	%DRx.S %DRx.F	当前步数 满	读 读
步进计数器	%SCx.n	步进计数器位	读/写
寄存器	%Rx.I %Rx.O %Rx.E %Rx.F	输入 输出 空 满	读/写 读 读/写 读/写
移位寄存器	%SBR.x.yy	寄存器位	读/写
消息	%MSGx.D %MSGx.E	完成 错误	读 读
AS-i 从设备输入	%IA.x.y.z	值	读/强制
AS-I 模拟从设备输入	%IWA.x.y.z	值	读
AS-i 从设备输出	%QA.x.y.z	值	读/写/强制
AS-I 模拟从设备输出	%QWA.x.y.z	值	读/写

注意：

1．变量如果没有在程序中使用，将不会被显示，因为 Twido 使用动态存储分配。

2．如果%MW的值大于+32767或小于-32768，操作显示将连续闪烁。

3．如果%SW的值大于65535，操作显示将连续闪烁，除了%SW0和%SW11。如果输入值超出限制，其值将返回到配置值。

4．如果%PLS.P的输入值超出限制，其值将写为饱和值。

对象和变量显示与修改

每个系统对象类型通过开始输入对象(%I)被访问，顺序经过消息对象(%MSG)，最后环回到输入对象(%I)。

显示一个系统对象：

步骤	动作
1	按下▶键直到数据显示屏幕出现。 输入对象("I")将被显示在显示区域的左上角。字母" I "（或者前面查看数据对象的名字）不闪烁。
2	按下 MOD/ENTER 键进入编辑模式。 输入对象"I"字符（或者前面查看数据对象的名字）开始闪烁。
3	按下▲键顺序移动对象列表。
4	按下▶键顺序移动对象类型并按下▲键增加其字段值。您能使用▶键和▲键操纵和修改显示对象的所有字段值。
5	重复步骤 3 和 4 直到编辑完成。
6	按下MOD/ENTER键接受修改值。 注意：对象的名字和地址在接受任何修改之前必须有效。即它们必须在使用操作显示之前就存在于控制器的配置中。 按下ESC放弃编辑模式下的任何修改。

数据值和显示格式

对象或变量的数据值通常以符号或非符号整数显示在显示区域的右下角。另外，所有字段禁止以零开头的显示值。操作显示中每个对象的地址以下面七种格式之一被显示：

- I/O 格式
- AS-I 从设备 I/O 格式
- 功能模块格式
- 简单格式
- 网络 I/O 格式
- 步进计数器格式
- 移位寄存器格式

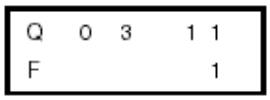
输入/输出格式

输入/输出对象（%I, %Q, %IW 和 %QW）具有三部分地址（例如：%IX.Y.Z）且被显示如下：

- 左上为对象类型和控制器地址
- 上中为扩展地址
- 右上为 I/O 通道

在简单输入(%I)和输出(%Q)的情况下，显示的左下部分包含一个字符，或者是"U"表示非强制或者是"F"表示强制位。强制值显示在屏幕的右下部分。

输出对象%Q0.3.11在显示区域显示如下：



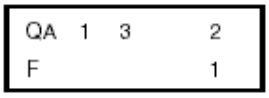
AS-I 从设备 I/O 格式

AS-i 从设备 I/O 对象(%IA, %QA, %IWA 和 %QWA)具有四部分地址（例如：%IAx.y.z）且被显示如下：

- 左上为对象类型
- 上左偏中为扩展总线上的 AS-i 主模块地址
- 上右偏中为 AS-i 总线上的从设备地址
- 右上为从设备 I/O 通道

在简单输入(%IA)和输出(%QA)的情况下，显示的左下部分包含一个字符，或者是"U"表示非强制或者是"F"表示强制位。强制值显示在屏幕的右下部分。

输出对象%QA1.3.2 在显示区域显示如下：

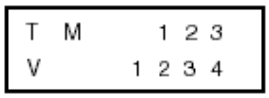


功能模块格式

功能模块(%TM, %C, %FC, %VFC, %PLS, %PWM, %DR, %R, 和 %MSGj)具有两部分地址，包含一个对象号和一个变量或属性名字。它们被显示如下：

- 左上为功能模块名字
- 右上为功能模块号（或例子）
- 左下为变量或属性
- 右下为属性值

下面示例中，123 号定时器当前值被设为 1,234。

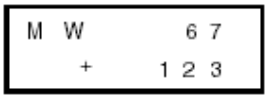


简单格式

简单格式用于对象%M, %MW, %KW, %MD, %KD, %MF, %KF, %S, %SW 和 %X：

- 右上为对象号
- 下面是对象的符号值

下面示例中，67号存储字包含值+123。

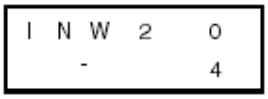


网络输入/输出格式

网络输入/输出对象(%INW 和 %QNW)在显示区域显示如下：

- 左上为对象类型
- 上中为控制器地址
- 右上为对象号
- 下面为对象符号值

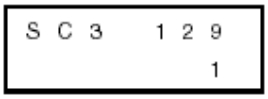
下面示例中，配置成远程地址#2 的远程控制器的第一个输入网络字被设为值-4。



步进计数器格式　步进计数器(%SC)格式显示对象号和步进计数器位如下：

- 左上为对象名字和编号
- 右上为步进计数器位
- 下面是步进计数器位的值

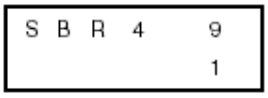
下面示例中，3号步进计数器的第129位被置为1。



移位寄存器格式　移位寄存器(%SBR)在显示区域显示如下：

- 左上为对象名字和编号
- 右上为寄存器位编号
- 右下是寄存器位值

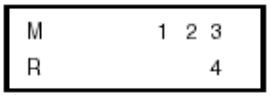
下面示例是4号移位寄存器的显示。



串口设置

导言

操作显示允许您显示协议设置并改变所有通过TwidoSoft配置的串口的地址。串口的最大数目是两个。下面示例中，第一个端口配置成Modbus协议，地址为23。第二个串口配置成远程连接，地址为。



串口设置显示和修改

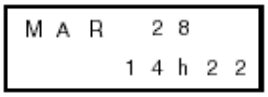
Twido 控制器最多可以支持两个串口。使用操作显示显示串口设置。

步骤	动作
1	按➡键直到通信显示出现。第一个串口协议设置的一个字母("M", "R", 或 "A")将显示在操作显示的左上角落。
2	按 MOD/ENTER 键进入编辑模式。
3	按➡键直到进入您要修改的字段。
4	按⬆键增加字段值。
5	继续 3 和 4 直到地址设置完成。
6	按 MOD/ENTER 键接受修改值或 ESC 放弃编辑模式中的任何修改。

日期时钟定时

导言

如果您的Twido控制器安装了RTC选件卡（TWDXCPRTC），您可以使用操作显示修改日期和时间。月被显示在HMI显示的左上方。月的地方将包含值"RTC"直到输入一个有效值。日显示在右上角。时刻用军事格式。小时和分显示在右下角并用字母"h"分开。下面示例显示RTC设置为三月28日下午2:22。



日期时钟时间显示和修改

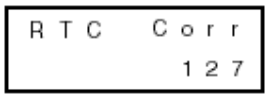
显示和修改日期时钟的时间：

步骤	动作
1	按➡键直到时间/日期显示出现。月值（“一月”，“二月”）将显示在显示区域的左上角。如果月没有被初始化，左上角将显示值"RTC"。
2	按 MOD/ENTER 键进入编辑模式。
3	按➡键直到进入您要修改的字段。
4	按⬆键增加字段值。
5	继续 3 和 4 直到日期时间值完成。
6	按 MOD/ENTER 键接受修改值或 ESC 放弃编辑模式中的任何修改。

实时修正因子

导言

您能使用操作显示显示和修改实时修正因子。每个实时时钟(RTC)选件模块有一个RTC修正因子，用于修正RTC模块晶体的不准确。修正因子是一个无符号3位整数从0到127，且显示在显示的右下角。
下面示例显示了修正因子127。



RTC 修正显示和修改

显示和修改实时修正因子：

步骤	动作
1	按➡键直到 RTC 因子显示出现。"RTC Corr"将显示在操作显示的上面一行。
2	按 MOD/ENTER 键进入编辑模式。
3	按➡键直到进入您要修改的字段。
4	按⬆键增加字段值。
5	继续 3 和 4 直到 RTC 修正值完成。
6	按 MOD/ENTER 键接受修改值或 ESC 放弃编辑模式中的任何修改。

概览

本部分的主题

本部分提供了梯形图，指令列表和 Grafcet 编程语言创建 Twido 可编程控制器控制程序的使用说明。

本部分包含了哪些内容？

本部分包含了以下章节：

11

概览

本章的主题　　本章描述了怎样使用梯形图语言编程。

本章包含了哪些内容？　本章包含了以下主题：

梯形图介绍

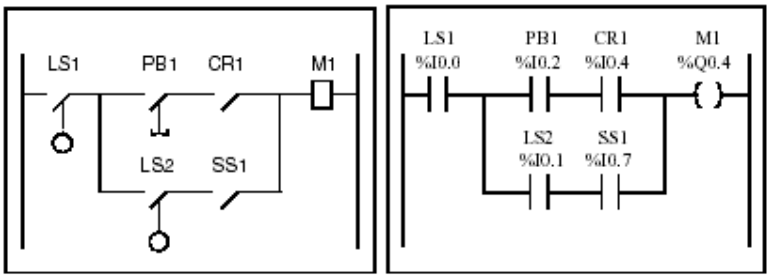
导言

梯形图类似于用来描述继电器电路的继电器逻辑图。两者之间的主要区别是继电器逻辑图没有梯形图下面的特点：

- 所有的输入都由触点符号（⊣⊢）表示。
- 所有的输出都由线圈符号（⊸⊢）表示。
- 梯形图指令中包括数字运算。

继电器逻辑电路等效梯形图

下面图例是一个继电器逻辑电路的简化接线图和他的等效梯形图。



继电器逻辑电路　　梯形图

请注意上面图例中，梯形图中所有与继电器逻辑图中开关设备相关的输入都以触点形式表示。继电器逻辑图中的 M1 输出线圈在梯形图中用输出线圈符号表示。梯形图中每个触点/线圈符号上的地址标号都对应于与控制器相连的外部输入/输出的位置。

梯级

用梯形图编写的程序由梯级构成，梯级是指画在象征电势的两条垂直栏里的图形指令集。梯级由控制器顺序执行。

图形指令集表述下述功能：

- 控制器的输入/输出（按钮，传感器，继电器，指示灯，等等）
- 控制器的功能（定时器，计数器，等等）
- 数学和逻辑运算（加法，除法，与，或，等等）
- 比较运算和其它数字运算（A<B，A=B，移位，循环，等等）
- 控制器的内部变量（位，字，等等）

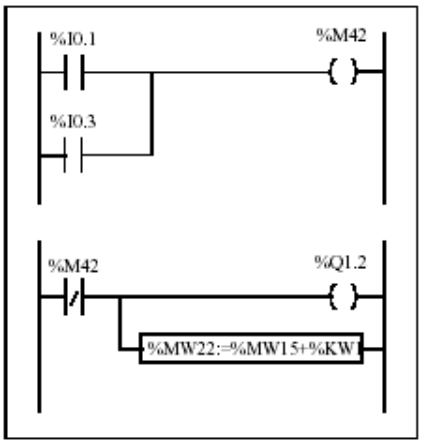
垂直和水平连接这些图形指令最终实现一个或多个输出和/或动作。一个梯级只能支持一组相关指令。

梯级示例

下图是一个由两个梯级组成的梯形图程序示例。

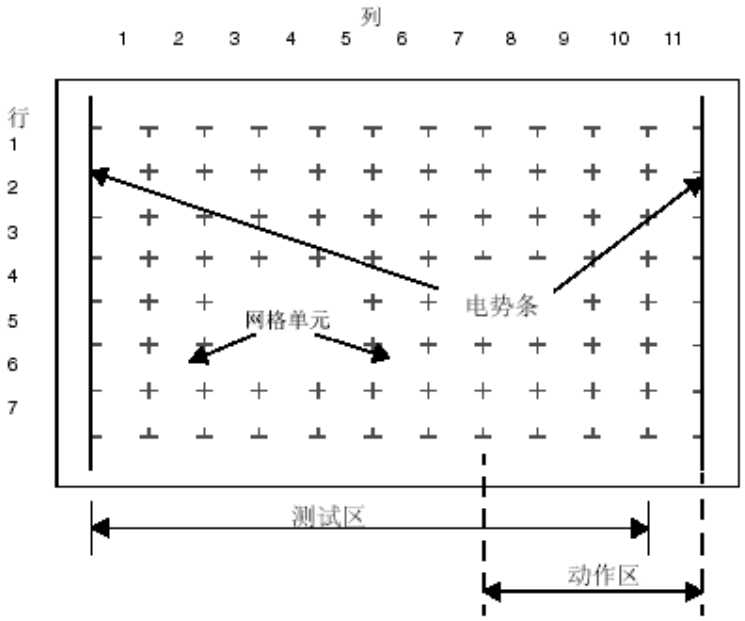
梯级 1 示例

梯级 2 示例



梯形图编程原则

编程网格　　每个梯级由 7 行 11 列组成，形成两个区域，如下图所示。



网格区域　　梯形图编程网格分为两个区：

- 测试区

包括动作发生所必须具备的条件。由列 1-10 组成，包括触点，功能模块，和比较模块组成。

- 动作区

包括测试区相关测试条件所引起的输出或操作。由列 8-11 组成并包括线圈和操作模块。

网格中指令输入

梯级提供了一个 7 行 11 列的编程网格，并从网格的最左上方单元开始。编程即是向网格中的单元输入指令。在测试区输入测试指令，比较模块，和功能模块，其中测试指令应该左对齐。测试逻辑为动作区提供了连贯性。在动作区里，线圈，数字运算，和程序流控制指令被输入并应该右对齐。梯级在网格中从上到下从左到右地被解释或执行（完成测试和赋值输出）。

梯级头

除了梯级以外，在它上方还有一个梯级头，用以说明梯级的逻辑目的。梯级头可以包括下列信息：

- 梯级编号
- 标号(%Li)
- 子程序说明(SRi:)
- 梯级标题
- 梯级注释

关于使用梯级头说明程序的更详细情况，见程序文档，p.227。

梯形图模块

导言

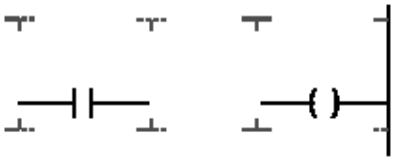
梯形图由下列表示程序流和功能的模块组成：

- 触点
- 线圈
- 程序流程指令
- 功能模块
- 比较模块
- 操作模块

触点，线圈和程序流程

触点，线圈，和程序流程（跳转和调用）指令占据梯形图编程网格的一个单元。功能模块，比较模块，和操作模块占据多个单元。

下面是触点和线圈示例。



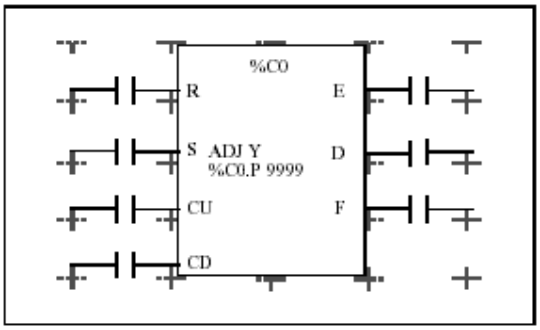
触点　　线圈

功能模块

功能模块位于编程网格的测试区。该模块必须放置在第一行；它的上面和下面不允许出现梯形图指令或连接线。梯形图测试指令导入功能模块的输入边，测试指令和/或动作指令由模块的输出边导出。

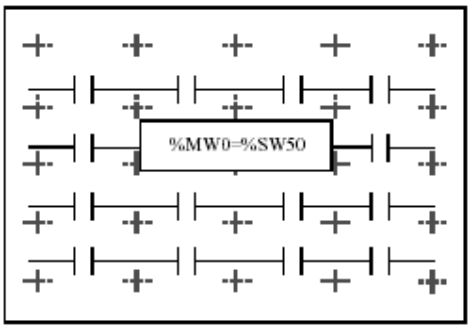
功能模块垂直指向，并占据编程网格的四行两列。

下面是一个计数器模块示例。



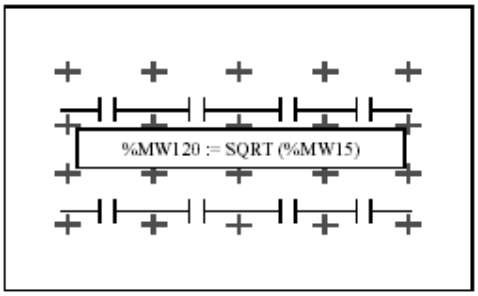
比较模块

比较模块位于网格中的测试区。它可以出现在测试区的任意行列，只要指令全长不超出测试区。
比较模块呈水平走向。它占据网格中的一行两列。
见下面比较模块示例。



操作模块

操作模块位于编程网格的动作区。该模块可以出现在动作区的任意一行。指令右对齐；它出现在右边直至最后一列。
操作模块呈水平走向且占据编程网格的一行四列。
下面是一个操作模块示例。



梯形图图形元素

导言　　梯形图指令由图形元素组成。

触点　　触点图形元素用于测试区编程且占据一个单元（一行一列）。

名字	图形元素	指令	功能
常开触点		LD	当控制位对象为状态 1 时通过触点。
常闭触点		LDN	当控制位对象为状态 0 时通过触点。
上升沿触点		LDR	上升沿：检测控制位对象从 0 变为 1。
下降沿触点		LDF	下降沿：检测控制位对象从 1 变为 0。

连接元素　　图形连接元素用于连接测试和动作图形元素。

名字	图形元素	功能
水平连接		两条电势栏之间测试和动作图形元素的连续连接。
垂直连接		平行测试和动作图形元素连接。

线圈

线圈图形元素用于动作区编程且占据一个单元（一行一列）。

名字	图形元素	指令	功能
直接线圈	-( )-	ST	相关位对象得到测试区结果值。
取反线圈	-(/)-	STN	相关位对象得到测试区结果的相反值。
置位线圈	-(S)-	S	当测试区结果为 1 时相关位对象被置为 1。
复位线圈	-(R)-	R	当测试区结果为 1 时相关位对象被置为 0。
跳转或子程序调用	->>%Li ->>%SRi	JMP SR	连接到一个标注指令。向上跳转或向下跳转。
转换条件线圈	-(#)-		Grafcet 语言。当编程与转换相关的条件时引起跳转到下一步时被使用。
主控继电器	-(MCS)- -(MCR)-	MCS MCR	修改一个程序的执行。
子程序返回	<RET>	RET	位于子程序末端返回到主程序。
结束程序	<END>	END	程序结束定义。

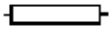
功能模块

功能模块的图形元素在测试区被编程且需要四行两列单元(除了超高速计数器需要五行两列)。

名字	图形元素	功能
定时器，计数器，寄存器，等等		每个功能模块使用输入和输出使得和其它图形元素连接。 注意：功能模块的输出不能互相连接(垂直短接）。

操作和比较模块

比较模块在测试区被编程，操作模块在动作区被编程。

名字	图形元素	功能
比较模块		比较两个操作数，当结果确认时输出变为1。 大小：一行两列
操作模块		完成算术和逻辑运算。 大小：一行四列

特殊梯形图指令 OPEN 和 SHORT

导言

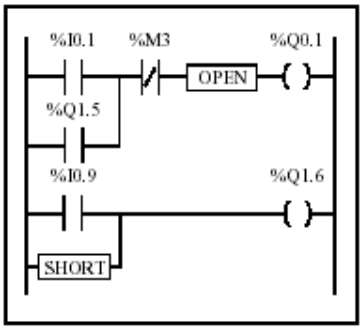
OPEN 和 SHORT 指令为梯形图程序的调试和故障排除提供了简便方法。这两个特殊指令通过短路或者开路梯级来改变一个梯级的逻辑，如下表所解释。

指令	描述	列表指令
OPEN	在梯级的连续性中创建一个断点，不管最近的逻辑运算结果。	AND 0
SHORT	允许梯级连贯地通过，不管最近的逻辑运算结果。	OR 1

列表编程中，OR和AND用于创建OPEN 和 SHORT 指令，分别使用立即值0和1。

示例

下面是 OPEN 和 SHORT 指令使用示例。



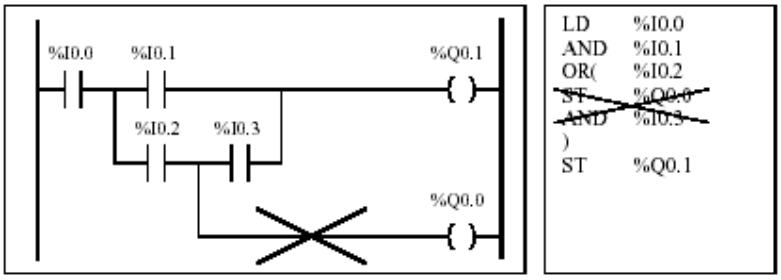

```
LD      %I0.1
OR      %Q1.5
ANDN    %M3
AND     0
ST      %Q0.1
LD      %I0.9
OR      1
ST      %Q1.6
```

编程建议

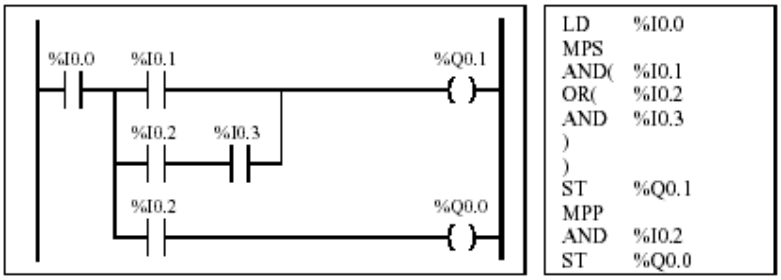
程序跳转处理	使用程序跳转应避免回路过长，否则会增加扫描时间。避免向上跳转。（向上跳转，标号位于跳转指令之前。 向下跳转，标号位于跳转指令之后。）
输出编程	输出位，类似的内部位，在程序中只能被修改一次。在输出位情形下，当输出被刷新时，只有最后的扫描值被保留。
使用直接连线的紧急停止传感器	传感器直接用于控制器不能处理的紧急停止。它们必须直接连接到对应输出。
电源返回处理	在手动操作条件下使电源返回。自动重启可能导致意外的设备操作（使用系统位%S0, %S1 和 %S9）。
时间和调度模块管理	应该检查系统位%S51 的状态，它显示任何 RTC 故障。
语法和错误检查	当输入程序时，TwidoSoft 检查指令，操作数，和它们之间结合的语法。

圆括号使用的另外一些注意

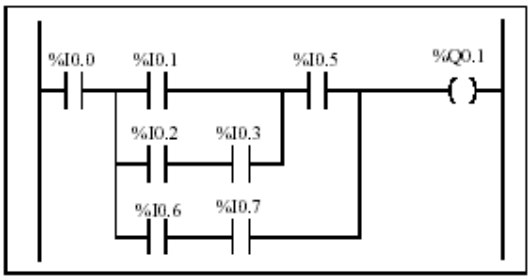
赋值操作不能位于圆括号内：

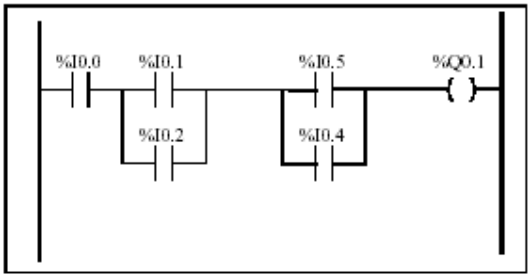


为了实现相同的功能，必须使用下面的等价编程：

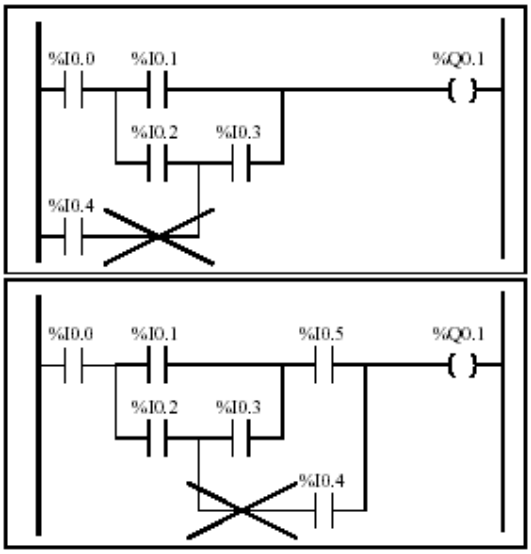


如果几个触点平行，它们必须互相嵌套或完全分开：

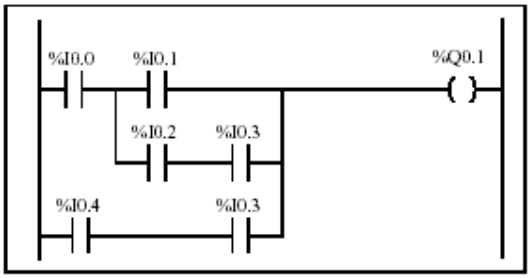




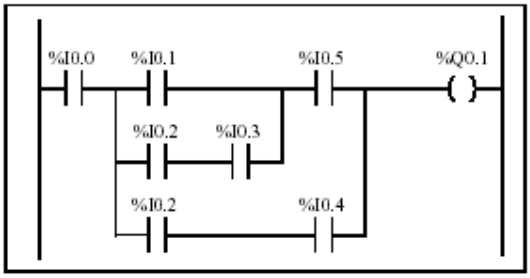
下面图表不能被编程：



为了执行它们的等价图表，它们必须修改如下：




```
LD      %I0.0
AND(    %I0.1
OR(     %I0.2
AND     %I0.3
)
)
OR(     %I0.4
AND     %I0.3
)
ST      %Q0.1
```




```
LD      %I0.0
AND(    %I0.1
OR(     %I0.2
AND     %I0.3
)
AND     %I0.5
OR(     %I0.2
AND     %I0.4
)
)
ST      %Q0.1
```

梯形图/指令列表可逆性

导言

程序可逆是 TwidoSoft 编程软件的一个特点，指支持应用程序在梯形图和指令列表之间的相互转换。

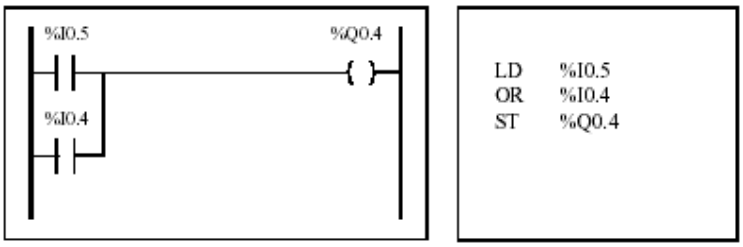
使用TwidoSoft设置程序的默认显示：指令列表或者梯形图格式（通过用户参数设置）。TwidoSoft也可以用来切换指令列表和梯形图视图。

可逆性理解

理解程序可逆特点的关键在于清楚梯级和相关指令列表之间的关系：

- 梯级：梯形图指令集合组成的逻辑表达式。
- 列表序列：列表编程指令集合，对应梯形图指令并表示相同逻辑的表达式。

下面图例显示了一个普通梯级和它的列表指令序列等效逻辑表达程序。



应用程序内部都是以列表指令形式存储，不管程序是通过梯形图语言还是列表语言写成。TwidoSoft 利用这两种语言间的程序结构相似性，使用程序内部列表映像将程序显示在列表和梯形图视图和编辑器里，或者是一个列表程序（它的基本形式），或者图形化为一个梯形图，这取决于用户参数的选择。

可逆性保证

梯形图程序一般可以转换到指令列表。然而，一些列表逻辑不能转换到梯形图。为了保证指令列表到梯形图的可逆性，需要遵守梯形图/指令列表可逆原则，p.225 中的列表编程原则。

可逆所需指令

列表语言中可逆功能模块的结构需要使用下列指令：

- **BLK** 标志模块开始，并定义梯级的开始和模块输入部分的起始。
- **OUT_BLK** 标志模块输出部分的起始。
- **END_BLK** 标志模块和梯级的结束。

指令列表程序不是一定要使用可逆功能模块才能正常运行。因为一些指令可以用于列表编程但不可逆。对于标准功能模块的不可逆列表编程的描述，见标准功能模块编程原则，p.280。

需要避免的不等价指令

避免使用某些没有等价梯形图的列表指令，或指令与操作数的结合。例如，指令 N（布尔累加器值取反）没有等价梯形图指令。

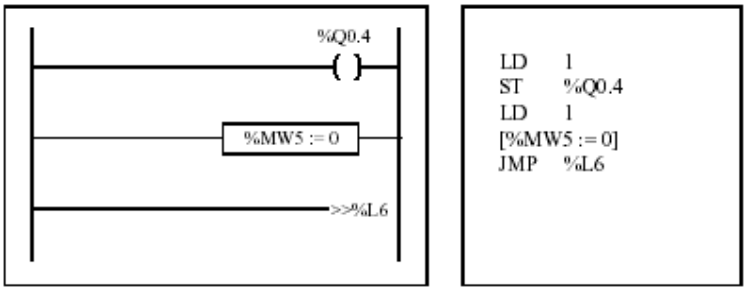
下表列出了所有不能转换到梯形图的列表编程指令。

列表指令	操作数	描述
JMPCN	%Li	条件为非时跳转
N	没有	取反（Not）
ENDCN	没有	条件为非时结束

无条件梯级

无条件梯级编程也需要遵守列表编程原则以保证列表和梯级之间的可逆性。无条件梯级没有测试或条件。输出或动作指令在任何情况下都能被激活或执行。

下图是无条件梯级示例及其等价列表序列。



注意除了 JMP 指令外，上面无条件列表的每个序列都以紧跟着数字 1 的装入指令开始。这个结合将布尔累加器置为 1，从而每次程序扫描时线圈（存储指令）都置为 1，%MW5 都置为 0。无条件跳转列表指令(JMP %L6)是例外，它的执行与累加器的值无关，不需要将累加器置为 1。

梯形列表栏

如果一个列表程序不完全可逆，则其可逆部分以梯形图显示，不可逆部分显示在梯形列表栏。

梯形列表栏功能类似一个小型列表编辑器，允许用户查看和修改梯形图程序的不可逆部分。

程序说明

您的程序说明

您能使用列表和梯形图编辑器输入注释来说明您的程序：

- 使用列表编辑器通过列表注释行来说明您的程序。这些注释可与编程指令同行，也可以另外起行。
- 使用梯形图编辑器通过梯级头来说明您的程序。它们位于梯级上方。

TwidoSoft编程软件使用这些可逆性注释。当一个程序从列表转换为梯形图时，TwidoSoft使用一些列表注释构建梯级头。因此，列表序列间插入的注释将用于梯级头。

列表注释行示例

下面是带有列表注释行的列表程序示例。

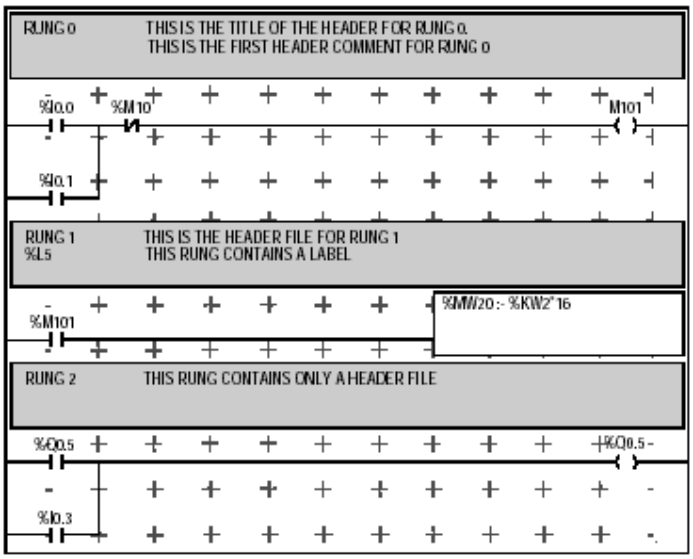
```
---- ( * THIS IS THE TITLE OF THE HEADER FOR RUNG 0 * )
---- ( * THIS IS THE FIRST HEADER COMMENT FOR RUNG 0 * )
---- ( * THIS IS THE SECOND HEADER COMMENT FOR RUNG 0 * )
     0   LD  %I0.0  ( * THIS IS A LINE COMMENT *)
     1   OR  %I0.1  ( * A LINE COMMENT IS IGNORED WHEN REVERSING TO LADDER * )
     2   ANDM  %M10
     3   ST        M101
---- ( * THIS IS THE HEADER FOR RUNG 1 * )
---- ( * THIS RUNG CONTAINS A LABEL * )
---- ( * THIS IS THE SECOND HEADER COMMENT FOR RUNG 1 * )
---- ( * THIS IS THE THIRD HEADER COMMENT FOR RUNG 1 * )
---- ( * THIS IS THE FOURTH HEADER COMMENT FOR RUNG 1 * )
     4   %L5:
     5   LD  %M101
     6   [%MW20 := %KW2 * 16 ]
---- ( * THIS RUNG ONLY CONTAINS A HEADER TITLE * )
     7   LD  %Q0.5
     8   OR  %I0.3
     9   ORR  I0.13
    10   ST  %Q0.5
```

列表注释到梯形图的转换

当列表指令转换到梯形图时，列表注释行根据下面规则显示在梯形图编辑器中：

- 单独起行的第一个注释用作梯级头标题。
- 第一个注释之后的所有注释成为梯级头的主体。
- 一旦梯级头存在主题注释，则可忽略列表之间的注释行，因为这类注释行包含了列表指令。

梯级头注释示例　下面是带有梯级头注释的梯形图程序示例。



梯形图注释到列表的转换　当梯形图转换到列表指令时，梯级头注释根据下面规则显示在列表编辑器中：

- 任何梯级头注释将被插入到相关列表序列之间。
- 任何标号(%Li:)或子程序声明(SRi:)位于标题的后面一行和列表序列之前。
- 如果列表以前曾转换到梯形图，则哪些被忽略的注释将重新出现在列表编辑器中。

12

概览

本章的主题　　本章描述了怎样使用指令列表语言编程。

本章包含了哪些内容？　　本章包含了以下主题：

列表程序概述

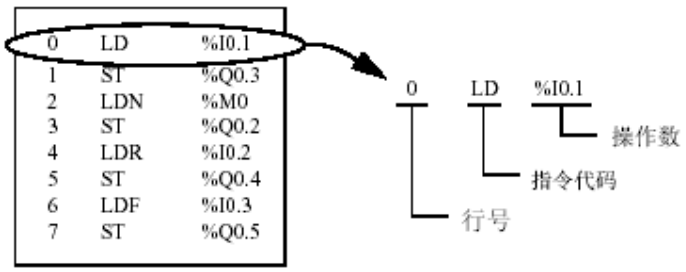
导言

由一系列指令组成的列表语言写成的程序被控制器顺序执行。每个列表指令表示成一个程序行并由三个部分组成：

- 行号
- 指令代码
- 操作数

列表程序示例

下面是一个列表程序示例。



行号

行号在您输入一个指令时自动生成。空行和注释行没有行号。

指令代码

指令代码是一种操作符号，用于确定使用操作数进行何种操作。典型操作特指布尔和数字运算。

例如，在上面示例程序中，LD 是 LOAD 指令代码的缩写。LOAD 指令放置（装载）操作数%I0.1 的值到一个内部寄存器，即累加器。

指令有两种基本形式：

- 测试指令

它们对动作执行的必要条件进行设置或测试。例如，LOAD (LD) 和 AND。

- 动作指令

它们按照设置条件的结果执行动作。例如，赋值指令如STORE (ST) 和 RESET (R)。

操作数

操作数是数字，地址，或符号，表示程序在指令中可以处理的一个值。例如，在上面的示例程序中，操作数%I0.1 是一个地址，其值是控制器的一个输入。一个指令根据指令代码的类型，可以有零到三个操作数。

操作数可以表示：

- 控制器输入和输出如传感器，按钮，和继电器。
- 预置系统功能如定时器和计数器。
- 算术，逻辑，比较和数字运算。
- 控制器内部变量如位和字。

列表指令操作

导言

列表指令只有一个显式操作数，另外是隐式操作数。隐式操作数是布尔累加器中的值。例如，指令 LD %I0.1 中，%I0.1 为显式操作数。一个隐式操作数存储于累加器中并将被%I0.1 的值覆盖。

操作

列表指令对累加器和显式操作数的内容执行指定的操作，且其结果将替代累加器。例如，执行 AND %I1.2 指令表示将累加器的值和输入%I1.2 进行逻辑与运算，并用运算结果替换累加器的内容。

所有布尔指令，除了Load, Store, 和 Not，都对两个操作数进行操作。两个操作数的值可以是True 或 False，且指令的程序执行产生一个值：True 或 False。Load指令将操作数的值装入累加器，Store指令将累加器的值传递给操作数，Not指令没有显式操作数，只是转换累加器的状态。

受支持的列表指令

下面表格显式了列表指令语言的指令选集：

指令类型	示例	功能
位指令	LD %M10	读内部位%M10
模块指令	IN %TM0	开始定时器%TM0
字指令	[%MW10 := %MW50+100]	加运算
程序指令	SR5	调用子程序#5
Grafcet 指令	-*-8	步#8

导言

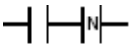
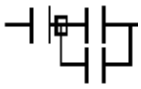
列表语言由下面几种语言组成：

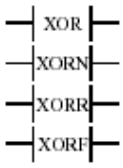
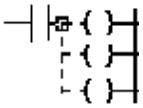
- 测试指令
- 动作指令
- 功能模块指令

本节辨别和描述了用于列表编程的Twido指令。

测试指令

下表描述了列表语言中的测试指令。

名字	等价梯形图元素	功能
LD		布尔运算结果与操作数状态相同。
LDN		布尔运算结果为操作数状态取反。
LDR		当检测到操作数（上升沿）从 0 变为 1 时布尔运算结果变为 1。
LDF		当检测到操作数（下降沿）从 1 变为 0 时布尔运算结果变为 1。
AND		布尔运算结果等于前面指令布尔运算结果和操作数状态的逻辑与结果。
ANDN		布尔运算结果等于前面指令布尔运算结果和操作数状态取反的逻辑与结果。
ANDR		布尔运算结果等于前面指令布尔运算结果和操作数上升沿（1=上升沿）检测的逻辑与结果。
ANDF		布尔运算结果等于前面指令布尔运算结果和操作数下降沿（1=下降沿）检测的逻辑与结果。
OR		布尔运算结果等于前面指令布尔运算结果和操作数状态的逻辑或结果。
AND(		逻辑与（8 层嵌套）
OR(		逻辑或（8 层嵌套）

名字	等价梯形图元素	功能
XOR, XORN, XORR, XORF		异或
MPS MRD MPP		转换到线圈
N	-	取反(NOT)

动作指令

下表描述了列表语言中的动作指令。

名字	等价梯形图元素	功能
ST	-( )-	相关操作数取值为测试区结果值。
STN	-(/)-	相关操作数取值为测试区结果值取反。
S	-(S)-	当测试区结果为1时相关操作数置为1。
R	-(R)-	当测试区结果为1时相关操作数置为0。
JMP	->>%Li	无条件向上或向下转移到一个标记序列。
SRn	->>%SRi	转移到子程序开始。
RET	<RET>	从子程序返回。
END	<END>	程序结束。
ENDC	<ENDC>	布尔运算结果为 1 时程序结束。
ENDCN	<ENDCN>	布尔运算结果为 0 时程序结束。

功能模块指令　　下表描述了列表语言中的功能模块指令。

名字	等价梯形图元素	功能
定时器，计数器，寄存器，等等。		每个功能模块均有模块控制指令。 一个结构化的格式直接用于连线模块的输入和输出。 注意：功能模块的输出不能互相连接（垂直短接）。

圆括号使用

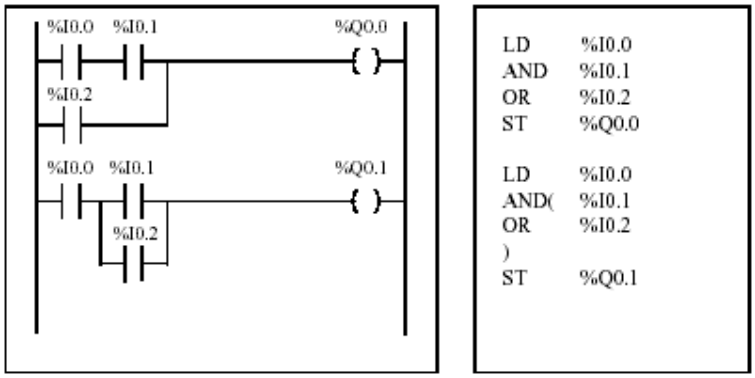
导言

在逻辑与和逻辑或指令中，圆括号用于指定梯形图的并列部分。圆括号和指令关联如下：

- 左括号与指令AND 或 OR相连。
- 右括号与每个左括号后的指令相连。

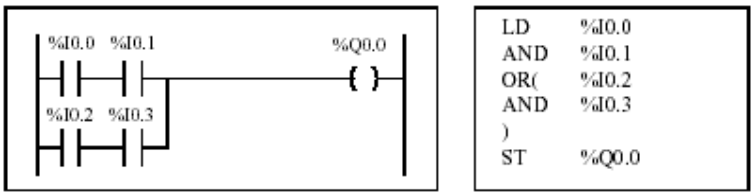
与指令使用示例

下图是圆括号和与指令使用示例：AND(...)。



或指令使用示例

下图是圆括号和或指令使用示例：OR(...)。



修饰符

下表列出了可用于圆括号的修饰符。

修饰符	功能	示例
N	取反	AND(N or OR(N
F	下降沿	AND(F or OR(F
R	上升沿	AND(R or OR(R
[	比较	见比较指令，p.311

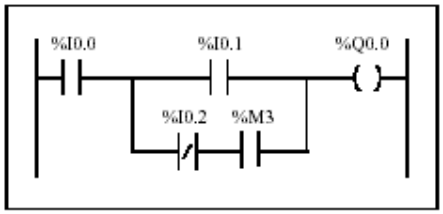
圆括号嵌套

最多可以嵌套八层圆括号。

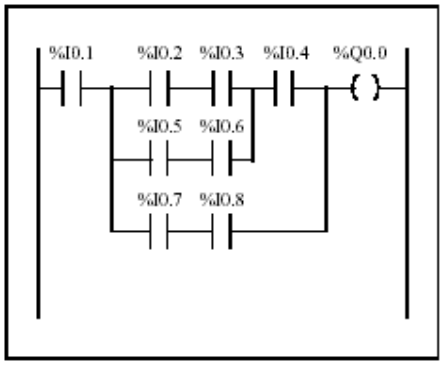
嵌套圆括号时请遵守下列规则：

- 左右括号必须对应。
- 标号(%Li:)，子程序(SRi:)，跳转指令(JMP)，和功能模块指令不能位于括号内。
- 存储指令ST, STN, S, 和 R不能位于括号内。
- 堆栈指令MPS, MRD, 和 MPP不能位于括号内。

圆括号嵌套示例　下图提供了圆括号嵌套示例。




```
LD      %I0.0
AND(    %I0.1
OR(N    %I0.2
AND     %M3
)
)
ST      %Q0.0
```




```
LD      %I0.1
AND(    %I0.2
AND     %I0.3
OR(     %I0.5
AND     %I0.6
)
AND     %I0.4
OR(     %I0.7
AND     %I0.8
)
)
ST      %Q0.0
```

堆栈指令(MPS, MRD, MPP)

导言

堆栈指令处理与线圈相关的线路。MPS, MRD, 和 MPP指令使用一个临时存储区即堆栈，可以存放最多八个布尔表达式。

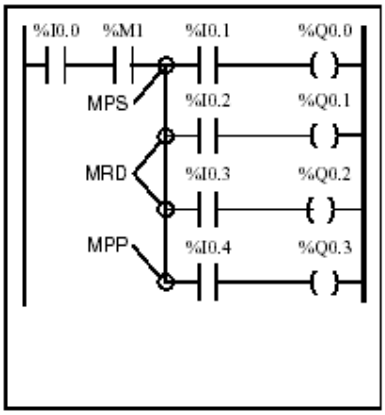
注意：这些指令不能用于圆括号内的表达式。

堆栈指令操作

下表描述了三个堆栈指令的操作。

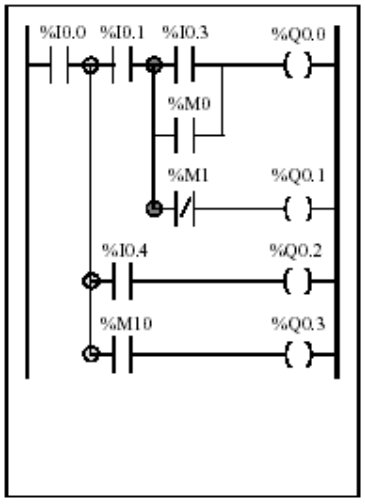
指令	描述	功能
MPS	存储压入堆栈	存储最近一次逻辑指令的结果（累加器的内容）到堆栈的顶部（压入），并使堆栈中其它值向堆栈底部移动一格。
MRD	从堆栈读取存储	将堆栈顶部值读入累加器。
MPP	从堆栈取出存储	将堆栈顶部值读入累加器（取出），并将堆栈内其它值向顶部移动一格。

堆栈指令示例　　下图是堆栈指令使用示例。




```
LD      %I0.0
AND     %M1
MPS
AND     %I0.1
ST      %Q0.0
MRD
AND     %I0.2
ST      %Q0.1
MRD
AND     %I0.3
ST      %Q0.2
MPP
AND     %I0.4
ST      %Q0.3
```

堆栈操作示例　　下图显式了堆栈指令怎样进行操作。




```
LD      %I0.0
MPS
AND     %I0.1
MPS
AND(    %I0.3
OR      %M0
)
ST      %Q0.0
MPP
ANDN    %M1
ST      %Q0.1
MRD
AND     %I0.4
ST      %Q0.2
MPP
AND     %M10
ST      %Q0.3
```

Grafcet

13

概览

本章的主题　　本章描述了怎样使用 Grafcet 语言编程。

本章包含了哪些内容？　　本章包含了以下主题：

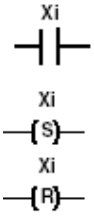
导言

TwidoSoft 中 Grafcet 指令提供了翻译控制序列的一个简单方法（Grafcet 表）。

Grafcet的最大步数取决于Twido控制器的型号。任何时刻活动步的数目仅由步的总数目所限制。对于TWDLCAA10DRF和TWDLCAA16DRF，可使用步1到62。步0和63保留为前处理和后处理。对所有其它控制器，可使用步1到95。

Grafcet 指令

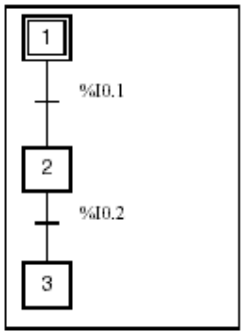
下表列出了Grafcet表编程所需的所有指令和对象：

图形表示（1）	TwidoSoft语言抄本	功能
图例：  初始步 转换 步	=*= i	开始初始步（2）
	# i	在停止当前步后激活步i
	-*- i	开始步i并使相关转换有效（2）
	#	停止当前步并不激活其它任何步
	#Di	停止步i和当前步
	=*= POST	开始后处理并结束顺序处理
	%Xi	步i的相关位，可以被测试和被写（步的最大数目取决于控制器）
	LD %Xi, LDN %Xi AND %Xi, ANDN %Xi, OR %Xi, ORN %Xi XOR %Xi, XORN %Xi	测试步i的活动性
	S %Xi	激活步i
	R %Xi	停止步i

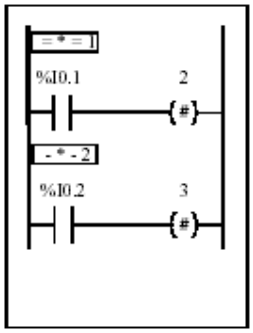
（1） 不考虑图形表示。

（2） 被写的第一步=*=i 或 -*-i表示顺序处理开始及预处理结束。

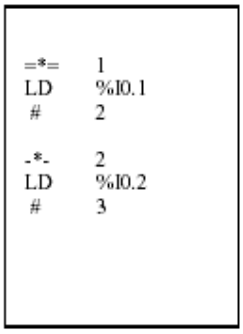
Grafcet 示例　　线性顺序：



不被支持

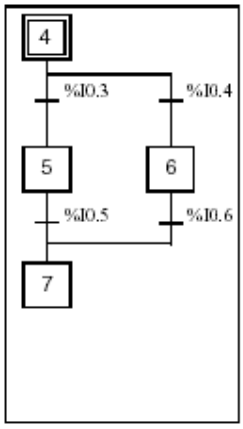


Twido梯形图

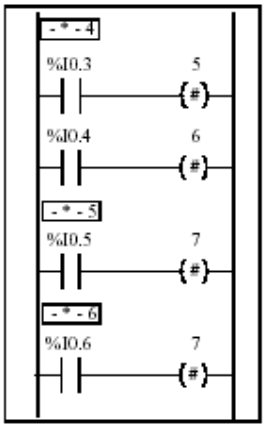


Twido列表程序

并列顺序：



不被支持



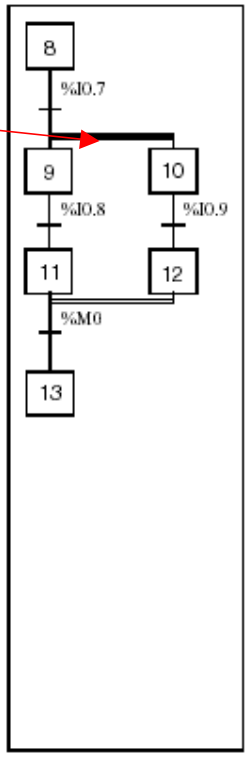
Twido梯形图

```
-*-     4
 LD     %I0.3
 #      5
 LD     %I0.4
 #      6

-*-     5
 LD     %I0.5
 #      7

-*-     6
 LD     %I0.6
 #      7
```

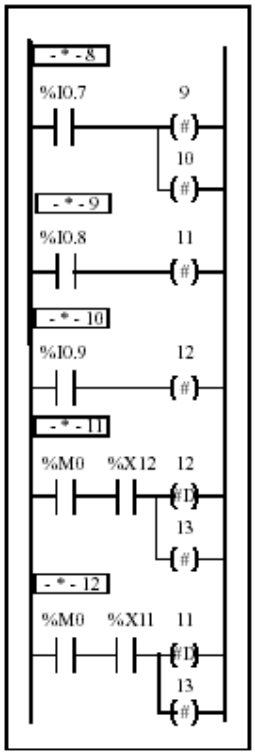
Twido列表程序

同步顺序：

双横线



不被支持



Twido梯形图

```
-*-     8
LD      %I0.7
#       9
#       10

-*-     9
LD      %I0.8
#       11

-*-     10
LD      %I0.9
#       12

-*-     11
LD      %M0
AND     %X12
#D      12
#       13

-*-     12
LD      %M0
AND     %X11
#D      11
#       13
```

Twido列表程序

注意：对于将要运行的 Grafcet 表，必须用=*=i 指令（初始步）声明至少一个活动步或在处理过程中使用系统位%S23 和指令 S %Xi 预置表。

Grafcet 程序结构描述

导言

一个TwidoSoft Grafcet 程序包含三部分：

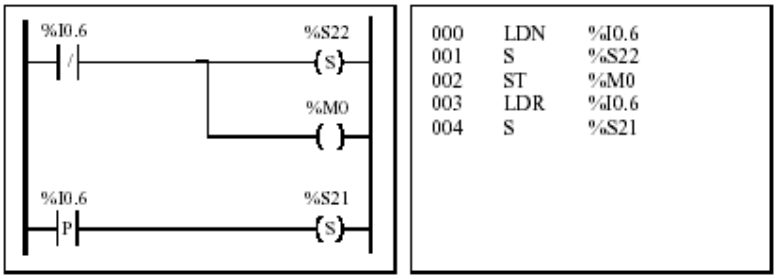
- 预处理
- 顺序处理
- 后处理

预处理

预处理由以下组成：

- 电源恢复
- 故障检查
- 修改工作模式
- 预置Grafcet步
- 输入逻辑

输入%I0.6的上升沿将位%S21置为1。它停止活动步并使非活动步能被激活。



预处理开始于程序的第一行，结束于"= * =" 或 "- * -"指令的第一次出现。三个系统位专用于Grafcet控制：%S21, %S22 和 %S23。它们通常在预处理时由程序置为1（如果需要）。预处理结束时系统执行其相应功能并将这些系统位复置到0。

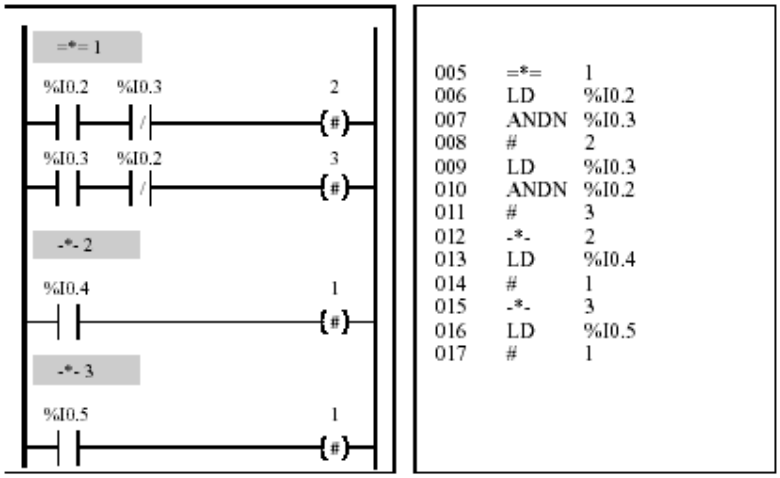
系统位	名字	描述
%S21	Grafcet 初始化	停止所有活动步并激活初始步。
%S22	Grafcet 重新初始化	停止所有步。
%S23	Grafcet 预置	如果对象%Xi 在预处理时被应用程序明确所写，则该位必须被置为1。如果该位由预处理保持为 1 且对象 %Xi 没有任何变化，则 Grafcet 被冻结（不考虑更新）。

顺序处理

顺序处理在表中发生（指令表示表）：

- 步
- 与步有关的动作
- 转换
- 转换条件

例如：



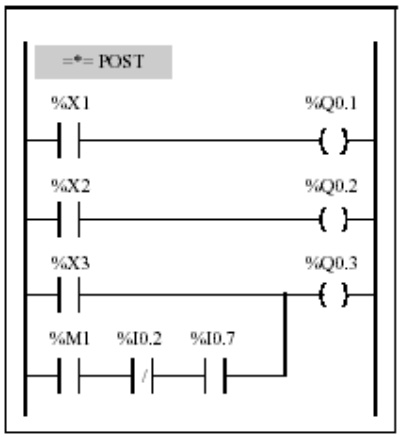
顺序处理结束于指令"= * = POST"的执行或程序结束。

后处理

后处理由如下组成：

- 来自顺序处理用于控制输出的命令
- 输出特定安全互锁

示例：




```
018     =*=     POST
019     LD      %X1
020     ST      %Q0.1
021     LD      %X2
022     ST      %Q0.2
023     LD      %X3
024     OR(     %M1
025     ANDN    %I0.2
026     AND     %I0.7
027     )
028     ST      %Q0.3
```

与 Grafcet 步相关的动作

导言

TwidoSoft Grafcet 程序提供了两种方法对与步相关的动作进行编程：

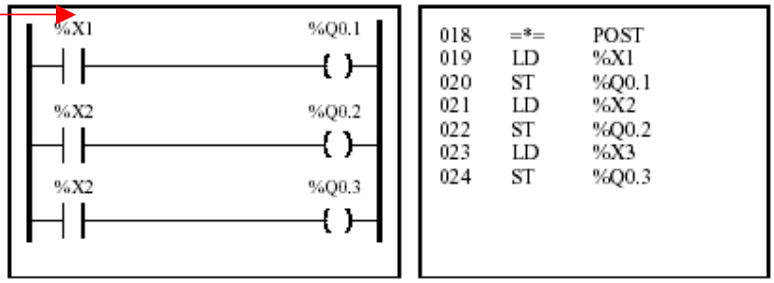
- 在后处理部分
- 在步自己的列表指令或梯级

在后处理中关联动作

如果存在安全或运行模式限制，最好在 Grafcet 应用程序的后处理部分对动作进行编程。您能通过 Set 和 Reset 列表指令或在梯形图程序中激活线圈来激活 Grafcet 步(%Xi)。

示例：

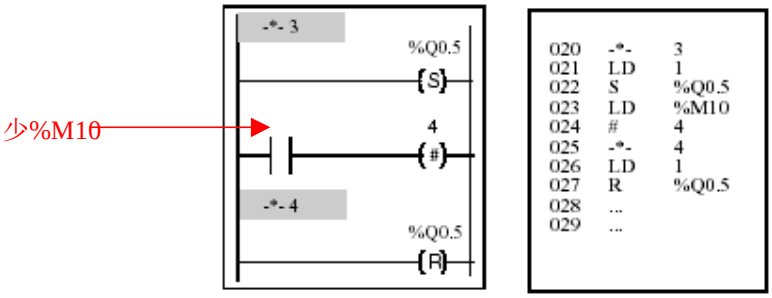
缺少 =*=（如上例）



用应用程序关联动作

您能在列表指令或梯级中对与步相关的动作进行编程。在这种情况下，列表指令或梯级不被扫描除非步处于活动状态。这是更为有效，可读，和可维持的使用Grafcet的方式。

示例：



少%M10

IV

概览

本部分的主题

本部分对 Twido 语言的基本和高级指令，以及系统位和字进行了详细描述。

本部分包含了哪些内容？

本部分包含了以下章节：

14

概览

本章的主题

本章提供了有关用于创建 Twido 控制器基本控制程序的指令和功能模块的详细资料。

本章包含了哪些内容?

本章包含了以下几节：

14.1 布尔运算

概览

本节的主题　　本节提供了布尔运算介绍，包括布尔指令描述和编程指导。

本节包含了哪些内容？　　本节包含了以下主题：

布尔指令

导言

布尔指令可与梯形图语言院院元素相比较。这些指令归纳如下表所示。

分类	指令	示例	描述
测试元素	装载 (LD) 指令等价于常开触点。	LD %I0.0	当位%I0.0 为 1 时触点闭合。
动作元素	存储 (ST) 指令等价于线圈。	ST %Q0.0	相关位对象取位累加器的逻辑值（前面逻辑运算的结果）。

测试元素的布尔运算结果应用于动作元素，如下面图例所示。

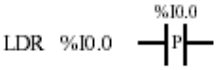
```
LD   %I0.0
AND  %I0.1
ST   %Q0.0
```

控制器输入测试

布尔测试指令可用于检测控制器输入的上升或下降沿。沿在第 n-1 次扫描时的输入状态与当前第 n 次扫描时的输入状态不同时被检测到。沿在当前扫描中保持检测值不变。

上升沿检测

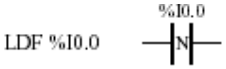
LDR 指令（装载上升沿）等价于上升沿检测触点。上升沿检测输入值从 0 到 1 的变化。正变换传感触点用于检测上升沿，如下图所示。



P：正变换传感触点

下降沿检测

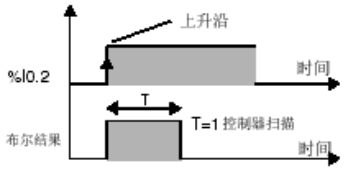
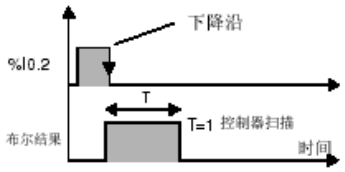
LDF 指令（装载下降沿）等价于下降沿检测触点。下降沿检测输入值从 1 到 0 的变化。负变换传感触点用于检测上升沿，如下图所示。



N：负变换传感触点

边沿检测

下表归纳了边沿检测的指令和时序：

边沿	测试指令	梯形图	时序图
上升沿	LDR %I0.0		
下降沿	LDF %I0.0		

边沿检测的内部位使用

上升或下降沿指令同样适用于内部位%M。

其余删除

注意：当发生冷或热重启时，即使输入保持为1，应用程序也会检测到一个上升沿。可以使用指令LD %S0, LD%S1, 和 ENDC作为程序的开始来消除这个现象。

布尔指令描述格式了解

导言

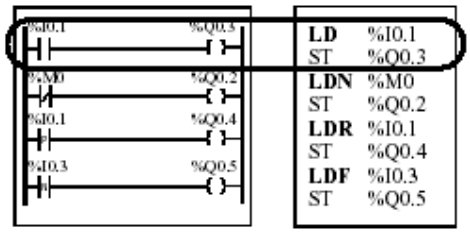
本节中的每个布尔指令通过下面几个方面的信息进行描述：

- 简述
- 指令示例及对应梯形图
- 允许操作数列表
- 时序图

下面解释对本节怎样描述布尔指令提供了更为详细的说明。

示例

下面图例显示了每个指令的示例是怎样给出的。



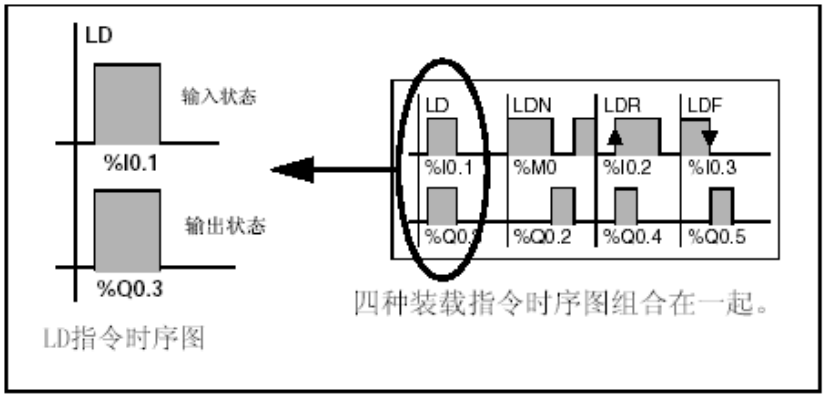
等价梯形图　　列表指令

允许操作数

下表定义了用于布尔指令的允许操作数类型。

操作数	描述
0/1	立即值 0 或 1
%I	控制器输入%Ii.j
%Q	控制器输出%Qi.j
%M	内部位%Mi
%S	系统位%Si
%X	步位%Xi
%BLK.x	功能模块位（例如，%TMi.Q）
%•:Xk	字位（例如，%MWi:Xk）
[	比较式（例如，[%MWi<1000]）

时序图　　下面图例显示了每个指令的时序图是怎样显示的。



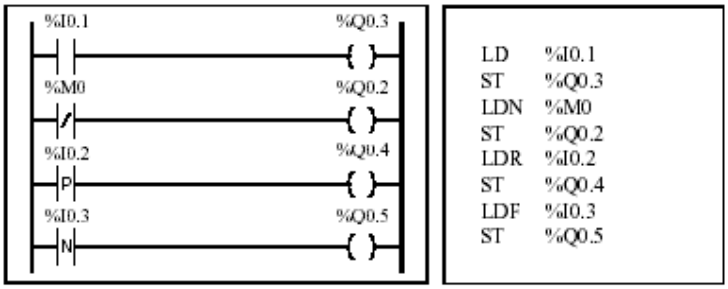
装载指令(LD, LDN, LDR, LDF)

导言

装载指令 LD, LDN, LDR，和 LDF 分别对应常开，常闭，上升沿，和下降沿（LDR 和 LDF 只用于控制器输入）。

示例

下图是装载指令示例。



允许操作数

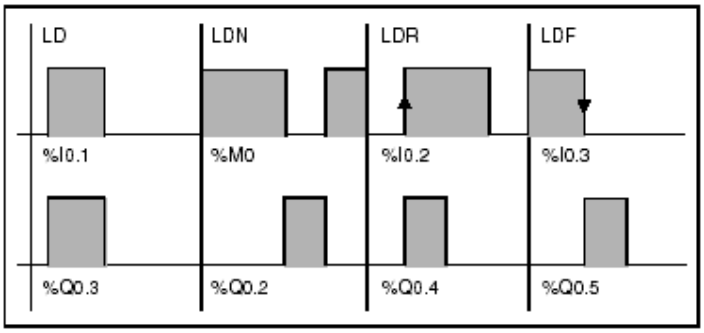
下表列出了装载指令类型，等价梯形图及允许操作数。

List Instruction	Ladder Equivalent	Permitted Operands
LD	⊣ ⊢	0/1,%I,%Q,%M,%S,%X,%BLK.x,%•:Xk,[
LDN	⊣/⊢	%I,%Q,%M,%S,%X,%BLK.x,%•:Xk,[
LDR	⊣P⊢	%I
LDF	⊣N⊢	%I

增加%M

增加%M

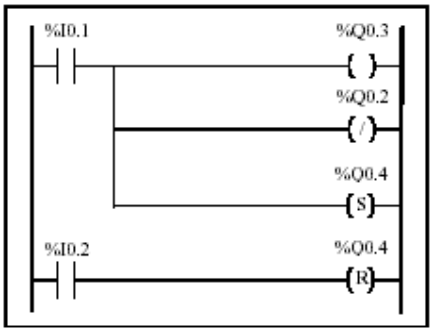
时序图　　下图显示了装载指令的时序图。



赋值指令(ST, STN, R, S)

导言　　赋值指令ST, STN, S, 和 R分别对应直接，反转，置位，和复位线圈。

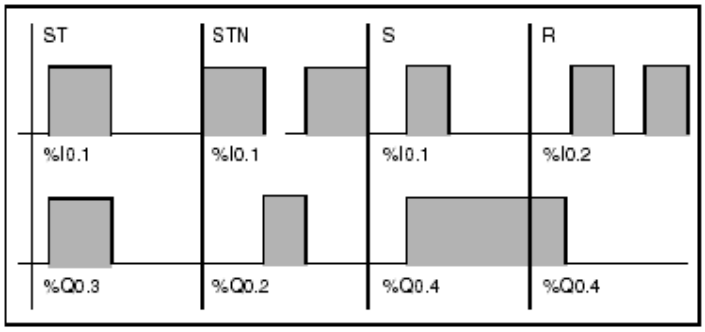
示例　　下图是赋值指令示例。



允许操作数　　下表列出了赋值指令类型，等价梯形图及允许操作数。

列表指令	等价梯形图	允许操作数
ST	{ }	%Q,%M,%S,%BLK.x,%•:Xk
STN	(/)	%Q,%M,%S,%BLK.x,%•:Xk
S	{S}	%Q,%M,%S,%X,%BLK.x,%•:Xk
R	{R}	%Q,%M,%S,%X,%BLK.x,%•:Xk

时序图　　下图显示了赋值指令的时序图。



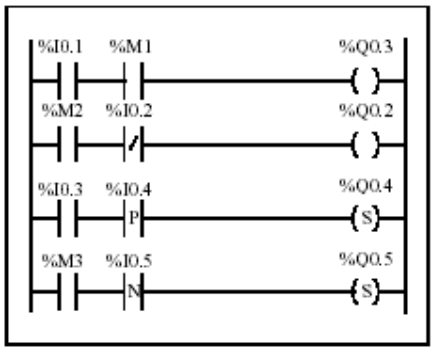
逻辑与指令(AND, ANDN, ANDR, ANDF)

导言

逻辑与指令执行操作数（或它的反转数，或上升沿，或下降沿）和前面指令的布尔运算结果间的逻辑与操作。

示例

下图是逻辑与指令示例。




```
LD      %I0.1
AND     %M1
ST      %Q0.3
LD      %M2
ANDN    %I0.2
ST      %Q0.2
LD      %I0.3
ANDR    %I0.4
S       %Q0.4
LD      %M3
ANDF    %I0.5
S       %Q0.5
```

允许操作数

下表列出了逻辑与指令类型，等价梯形图及允许操作数。

列表指令	等价梯形图	允许操作数
AND	( )	0/1,%I,%Q,%M,%S,%X,%BLK.x,%•:Xk, [
ANDN	(/)	%I,%Q,%M,%S,%X,%BLK.x,%•:Xk, [
ANDR	(S)	%I
ANDF	(R)	%I

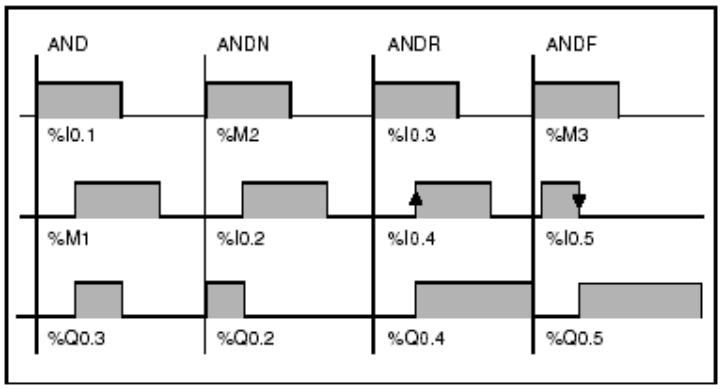
增加%M

增加%M

等价梯形图错误!

时序图

下图显示了逻辑与指令的时序图。



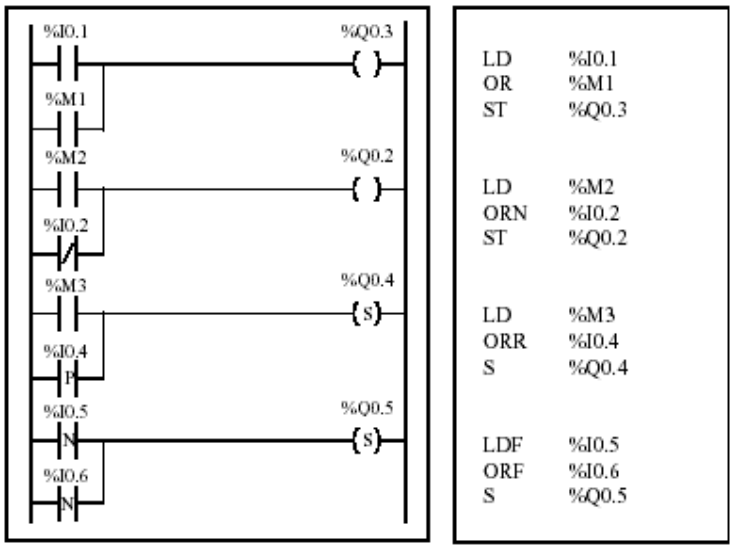
逻辑或指令(OR, ORN, ORR, ORF)

导言

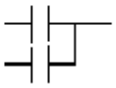
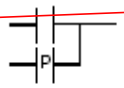
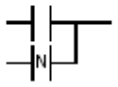
逻辑或指令执行操作数（或它的反转数，或上升沿，或下降沿）和前面指令的布尔运算结果间的逻辑或操作。

示例

下图是逻辑或指令示例。



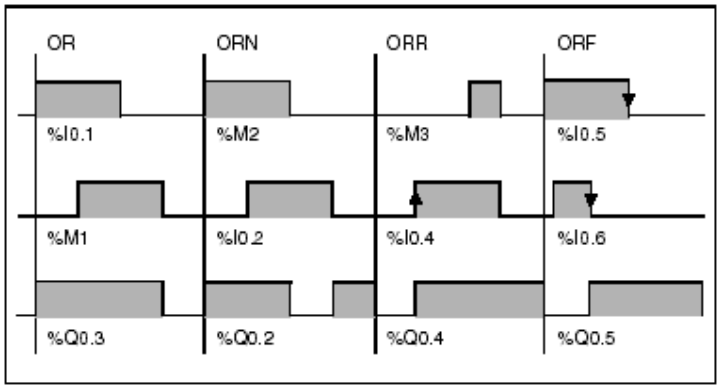
允许操作数　　下表列出了逻辑或指令类型，等价梯形图及允许操作数。

List Instruction	Ladder Equivalent	Permitted Operands
OR		0/1,%I,%Q,%M,%S,%X,%BLK.x,%•:Xk
ORN		%I,%Q,%M,%S,%X,%BLK.x,%•:Xk
ORR		%I
ORF		%I

增加%M

增加%M

时序图　　下图显示了逻辑或指令的时序图。



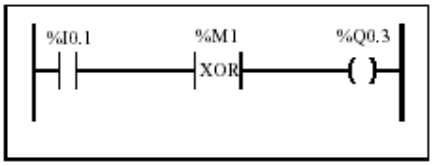
异或指令(XOR, XORN, XORR, XORF)

导言

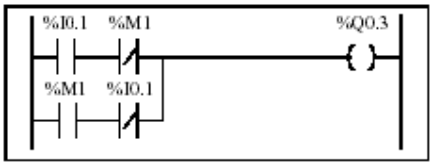
异或指令执行操作数（或它的反转数，或上升沿，或下降沿）和前面指令的布尔运算结果间的异或操作。

示例

下图是异或指令示例。




```
LD    %I0.1
XOR   %M1
ST    %Q0.3
```




```
LD    %I0.1
ANDN  %M1
OR(   %M1
ANDN  %I0.1
)
ST    %Q0.3
```

允许操作数

下表列出了异或指令类型及允许操作数。

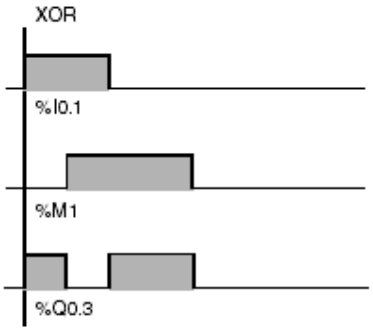
列表指令	允许操作数
XOR	%I,%Q,%M,%S,%X,%BLK.x,%•:Xk
XORN	%I,%Q,%M,%S,%X,%BLK.x,%•:Xk
XORR	%I
XORF	%I

增加%M

增加%M

时序图

下图显示了异或指令的时序图。

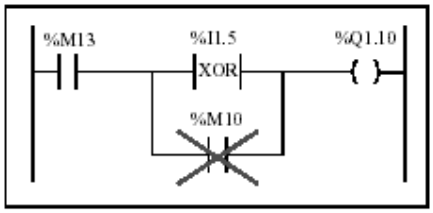


特殊情况

下面是对梯形图中使用异或指令的特别警告：

- 不要在梯级的第一个位置插入异或触点。
- 不要将异或触点与其它梯形图元素并行放置（见下面示例）。

如下图所示，加入一个与异或触点相并行的元素将会产生确认错误。



取反指令(N)

导言　　取反(N)指令将前面指令的布尔运算结果取反。

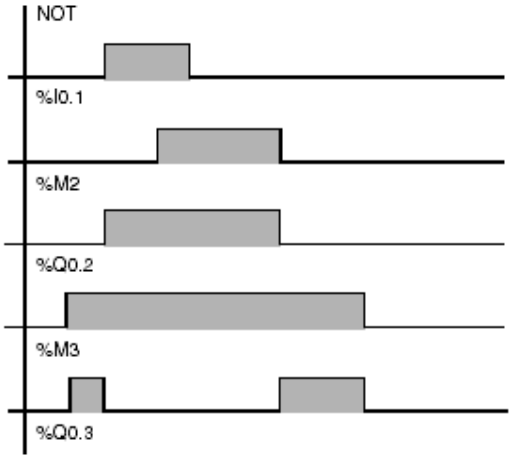
示例　　下图是取反指令使用示例。

```
LD    %I0.1
OR    %M2
ST    %Q0.2
N
AND   %M3
ST    %Q0.3
```

注意：取反指令不可逆。

允许操作数　　不适用。

时序图　　下图显示了取反指令的时序图。



14.2 基本功能模块

概览

本节的主题　　本节提供了基本功能模块使用描述和编程指导。

本节包含了哪些内容？　本节包含了以下主题：

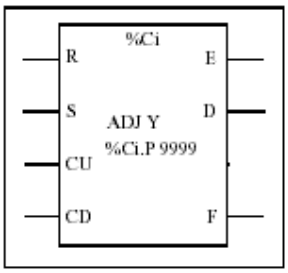
基本功能模块

导言

功能模块是程序使用的位对象和特殊字的来源。基本功能模块提供简单功能，如定时器或加/减计数。

功能模块示例

下图是一个加/减计数器功能模块示例。



加/减计数器模块

位对象

位对象对应模块输出。这些位可以通过下面某种方法由布尔测试指令访问：

- 直接访问（例如，LD E），如果它们在可逆化编程中与模块相连（见标准功能模块编程原则，p.280）。
- 通过指定模块类型（例如，LD %Ci.E）。

输入可以指令格式被访问。

字对象

字对象对应于指定参数和值：

- 模块配置参数：一些参数能被程序访问（如，预置参数），一些参数不能被程序访问（如，时基）。
- 当前值：例如，%Ci.V，当前计数值。

可访问位和字对象

下表描述了可被程序访问的基本功能模块位和字对象。

基本功能模块	符号	范围(i)	对象类型	描述	地址	写访问
定时器	%TMi	0 - 127	字	当前值	%TMi.V	不可以
				预置值	%TMi.P	可以
			位	定时器输出	%TMi.Q	不可以
加/减计数器	%Ci	0 - 31	字	当前值	%Ci.V	不可以
				预置值	%Ci.P	可以
			位	下溢输出（空）	%Ci.E	不可以
				达到预置输出	%Ci.D	不可以
				满溢输出（满）	%Ci.F	不可以

导言

用下面方法之一对标准功能模块编程：

- 功能模块指令（例如，BLK %TM2）：梯形图语言中这种可逆的编程方法使得运算可在程序的功能模块中执行。
- 特殊指令（例如，CU %Ci）：这种不可逆的方法使得运算在程序的几个模块输入处执行（例如，line 100 CU %C1, line 174 CD %C1, line 209 LD %C1.D）。

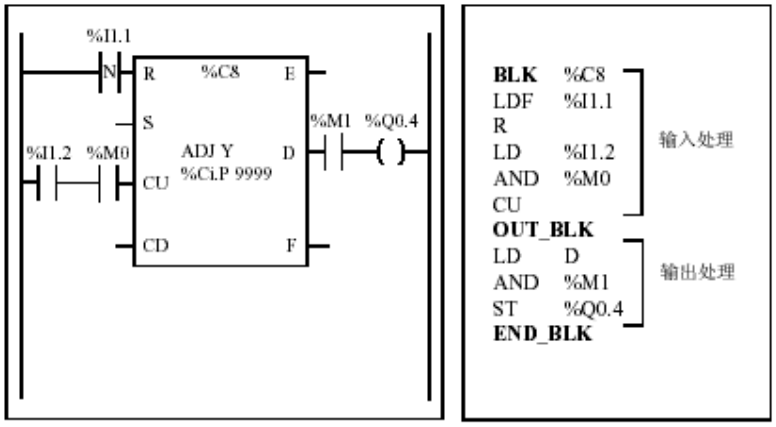
可逆编程

使用可逆编程指令 BLK, OUT_BLK, 和 END_BLK：

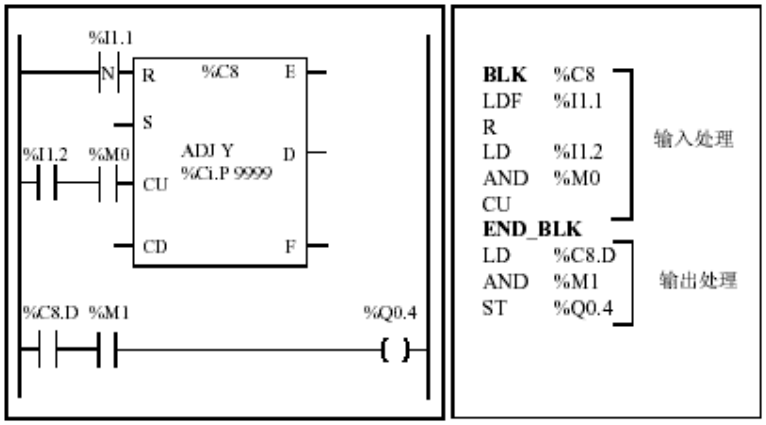
- **BLK:** 表示模块的开始
- **OUT_BLK:** 用于与模块输出直接连线。
- **END_BLK:** 表示模块的结束。

输出连线示例

下面示例显示了带有输出连线的计数器功能模块的可逆编程。



输出不连线示例　下面示例显示了不带有输出连线的计数器功能模块的可逆编程。



注意：只有相关模块中的测试和输入指令可以放在 BLK 和 OUT_BLK 指令之间（如果程序中没有 OUT_BLK 就放在 BLK 和 END_BLK 之间）。

定时器功能模块(%TMi)

导言

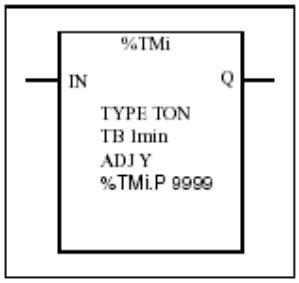
有三种定时器功能模块：

- TON（定时器 导通－延时）：这种定时器用于控制导通－延时动作。
- TOF（定时器 关断－延时）：这种定时器用于控制关断－延时动作。
- TP（定时器－脉冲）：这种定时器用于产生精确宽度的脉冲。

延时或脉冲周期可编程并可使用 TwidoSoft 进行修改。

图例

下面是定时器功能模块图例。



定时器功能模块

参数

定时器具有如下参数：

参数	标识	值
定时器编号	%TMi	0 到 63： TWDLCAA10DRF 和 TWDLCAA16DRF 0 到 127 对所有其它控制器。
类型	TON	I 定时器 导通－延时（默认）
	TOF	I 定时器 关断－延时
	TP	I 脉冲（单稳态）
时基	TB	1 min (默认), 1 s, 100 ms, 10 ms, 1 ms
当前值	%TMi.V	当定时器工作时，该字从 0 增加到 %TMi.P。可被程序读和测试，但不可写。%TMi.V 可以通过动态数据表编辑器修改。
预置值	%TMi.P	0 – 9999。该字可由程序读取，测试和写入。默认值是 9999。周期或产生的延时为 %TMi.P x TB。
动态数据表编辑器	Y/N	Y: Yes，预置%TMi.P值可以通过动态数据表编辑器修改。 N: No，预置%TMi.P 值不能被修改。
输入使能（或指令）	IN	定时器以上升沿（TON 或 TP 类型）或下降沿（TOF 类型）开始。
定时器输出	Q	根据实现的功能：TON, TOF, 或 TP，相关位%TMi.Q 置为 1。

注意：预置值越大，定时器的精度越高。

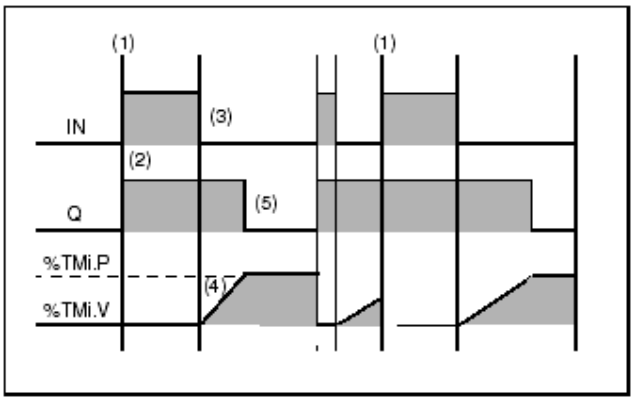
TOF 类型定时器

导言

使用 TOF (定时器关断－延时) 类型定时器控制关断－延时动作。延时可用 TwidoSoft 编程。

时序图

下面是 TOF 类型定时器操作时序图。



操作

下表描述了 TOF 类型定时器的操作。

阶段	描述
1	当前值%TMi.V在输入IN的上升沿置为0，即使定时器处于工作状态。
2	检测到输入 IN 的下降沿时%TMi.Q 输出位置为 1。
3	定时器在输入 IN 的下降沿开始工作。
4	当前值%TMi.V以时基TB为单位增加到%TMi.P。
5	当当前值到达%TMi.P 时%TMi.Q 输出位置为 0。

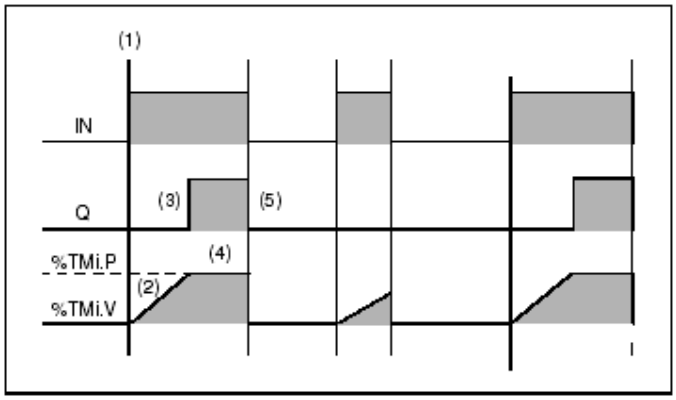
TON 类型定时器

导言

使用 TON (定时器导通－延时) 类型定时器控制导通－延时动作。延时可用 TwidoSoft 编程。

时序图

下面是 TON 类型定时器操作时序图。



操作

下表描述了 TON 类型定时器的操作。

阶段	描述
1	定时器在输入IN的上升沿开始工作。
2	当前值%TMi.V 以时基 TB 为单位增加到%TMi.P。
3	当当前值到达%TMi.P 时%TMi.Q 输出位置为 1。
4	当输入IN为1时%TMi.Q输出位保持为1。
5	当检测到输入 IN 的下降沿，定时器停止，即使定时器没有到达%TMi.P，且%TMi.V 置为 0。

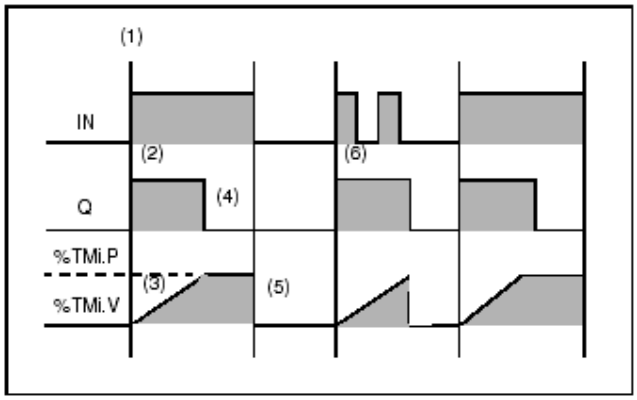
TP 类型定时器

导言

使用 TP (定时器－脉冲) 类型定时器用于产生具有精确宽度的脉冲。延时可用 TwidoSoft 编程。

时序图

下面是 TP 类型定时器操作时序图。



操作

下表描述了 TP 类型定时器的操作。

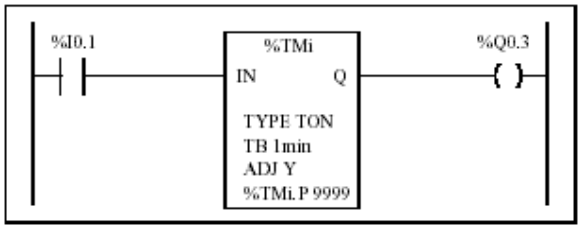
阶段	描述
1	定时器在输入IN的上升沿开始工作。如果定时器还没开始则当前值%TMi.V置为0。
2	当定时器开始时%TMi.Q 输出位置为 1。
3	当前值%Tmi.V 以时基 TB 为单位 0 增加到%Tmi.P。
4	当当前值到达%Tmi.P时%Tmi.Q输出位置为0。
5	当%TMi.V 等于%TMi.P 且输入 IN 回到 0 时当前值%Tmi.V 置为 0。
6	定时器不能被复位。一旦%TMi.V 等于%TMi.P 且输入 IN 为 0，则%Tmi.V 置为 0。

导言

不管定时器功能模块(%TMi)用途如何，它们的编程方法相同。定时器功能(TON, TOF, 或 TP)在配置中选定。

示例

下图是定时器功能模块可逆和不可逆编程示例。



可逆编程

```
BLK     %TM1
LD      %I0.1
IN
OUT_BLK
LD      Q
ST      %Q0.3
END_BLK
```

不可逆编程

```
LD      %I0.1
IN      %TM1
LD      %TM1.Q
ST      %Q0.3
```

配置

下面参数必须在配置中输入：

- 定时器类型：TON, TOF, 或 TP
- 时基：1 min, 1 s, 100 ms, 10 ms 或 1 ms
- 预置值(%TMi.P)：0 到 9999
- 调节：选中或不选中

特殊情况

下表是定时器功能模块编程特殊情况列表。

特殊情况	描述
冷重启(%S0=1)的影响	强制当前值为 0。输出%TMi.Q 置为 0。预置值复位到配置中的定义值。
热重启(%S1=1)的影响	对定时器的当前值和预置值都没有影响。电源损耗时当前值不变。
控制器停止的影响	停止控制器不会冻结当前值。
程序跳转的影响	跳过定时器模块并不冻结该定时器。定时器将继续增加直到达到预置值(%TMi.P)。此时，定时器模块赋值给输出 Q 的完成位(%TMi.Q)改变状态。然而，直接连线到模块输出的相关输出不被激活，并不被控制器扫描。
位%TMi.Q（完成位）测试	程序中最好只进行一次位%TMi.Q 测试。
主机控制继电器 指令 MCS/MCR 的影响	当指令 MCS 被激活时，两个 MCS/MCR 指令间编程的定时器模块被复位。
修改预置值%TMi.P 的影响	只有当定时器被重新激活时，通过指令修改或调节的预置值才起作用。

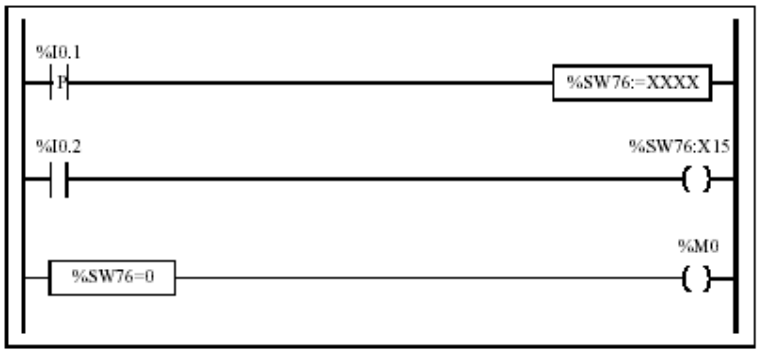
1 ms 时基的定时器

1 ms 时基只适用于前五个定时器。四个系统字%SW76, %SW77, %SW78, 和 SW79 可用作“沙漏”。这四个字如果具有正值，则系统对它们分别以毫秒为单位进行递减。

通过连续装载这些字中的一个或测试中间值可以得到多重定时。如果这四个字中一个值小于0，它将不会被修改。定时器可以通过设置对应的第15位为1“被冻结”，然后通过将其复位到0“解冻”。

编程示例　　下面是定时器功能模块编程示例。

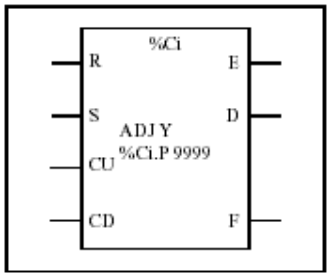
```
LDR     %I0.1          (Launching the timer on the rising edge of %I0.1)
[%SW76:=XXXX]          (XXXX = required value)
LD      %I0.2          (optional management of freeze, input I0.2 freezes)
ST      %SW76:X15
LD      [%SW76=0]      (timer end test)
ST      %M0
.....
```



加/减计数器功能模块(%Ci)

导言　　计数器功能模块(%Ci)提供事件的加和减计数。这两种运算可以同时进行。

图例　　下面是加/减计数器功能模块图例。



加/减计数器功能模块

参数

计数器功能模块具有如下参数：

参数	标识	值
计数器编号	%Ci	0 到 31
当前值	%Ci.V	字根据输入（或指令）CU 和 CD 被增加或减少。可被程序读和测试，但不可写。可使用数据编辑器修改%Ci.V。
预置值	%Ci.P	0 <= %Ci.P <= 9999。字可被读，测试，和写（默认值：9999）。
用动态数据表编辑器编辑	ADJ	▪ Y: Yes，预置值可以通过动态数据表编辑器修改。 ▪ N: No，预置值不能使用动态数据表编辑器修改。
复位输入（或指令）	R	状态为 1：%Ci.V = 0。
设置输入（或指令）	S	状态为 1：%Ci.V = %Ci.P。
加运算输入（或指令）	CU	在上升沿增加%Ci.V。
减运算输入（或指令）	CD	在上升沿减少%Ci.V。
减运算溢出输出	E（空）	当减计数器%Ci.V 从 0 变到 9999 时，相关位%Ci.E=1（当%Ci.V 到达 9999 时置为 1，如果计数器继续减少则复位为 0）。
预置输出达到	D（完成）	当%Ci.V=%Ci.P 时，相关位%Ci.D=1。
加运算溢出输出	F（满）	当%Ci.V 从 9999 变到 0 时，相关位%Ci.F=1（当%Ci.V 到达 0 时置为 1，如果计数器继续增加则复位为 0）。

操作

下表描述了加/减计数器操作的主要过程。

操作	动作	结果
加计数	加计数输入 CU 出现上升沿（或指令 CU 被激活）。	当前值%Ci.V 加 1。
	当前值%Ci.V 等于预置值%Ci.P。	“预置达到”输出位%Ci.D 变为 1。
	当前值%Ci.V 从 9999 变为 0。	输出位%Ci.F（加计数溢出）变为 1。
	如果计数器继续增加。	输出位%Ci.F（加计数溢出）复位到 0。
减计数	减计数输入 CD 出现上升沿（或指令 CD 被激活）。	当前值%Ci.V 减 1。
	当前值%Ci.V 从 0 变为 9999。	输出位%Ci.E（减计数溢出）变为 1。
	如果计数器继续减少。	输出位%Ci.E（减计数溢出）复位到 0。
加/减计数	要同时使用加计数和减计数功能（或同时激活指令 CD 和 CU），必须对同时控制两个对应的输入 CD 和 CU。这两个输入被连续扫描，如果它们都为 1，则当前值保持不变。	
复位	输入 R 置为状态 1（或者指令 R 被激活）。	当前值%Ci.V 被强制为 0。输出 %Ci.E, %Ci.D 和 %Ci.F 置为 0。复位输入优先。
置位	如果输入 S 置为 1（或指令 S 被激活）且复位输入为 0（或指令 R 未被激活）。	当前值%Ci.V 取%Ci.P 值且输出%Ci.D 置为 1。

特殊情况

下表显示了计数器特殊操作/配置情况列表。

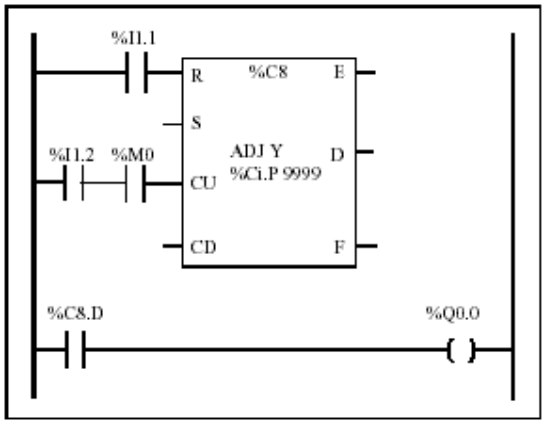
特殊情况	描述
冷重启(%S0=1)的影响	▌ 当前值%Ci.V置为0。 ▌ 输出位%Ci.E, %Ci.D, 和%Ci.F置为0。 ▌ 预置值根据配置定义值初始化。
控制器停止后热重启(%S1=1)的影响	对计数器的当前值(%Ci.V)没有影响。
修改预置%Ci.P的影响	当模块被应用程序处理（一个输入被激活）时，通过指令或调节来修改预置值会产生影响。

导言

下面示例是一个提供高达5000条计数的计数器。输入%I1.2的每个脉冲（当内部位%M0置为1时）都使计数器%C8增加，直至达到它的预置值（位%C8.D=1）。计数器的值由输入%I1.1复位。

编程示例

下图是计数器功能模块可逆编程和不可逆编程示例。



梯形图

```
BLK    %C8
LD     %I1.1
R
LD     %I1.2
AND    %M0
CU
END_BLK
LD     %C8.D
ST     %Q0.0
```

可逆编程

```
LD     %I1.1
R      %C8
LD     %I1.2
AND    %M0
CU     %C8
LD     %C8.D
ST     %Q0.0
```

不可逆编程

配置

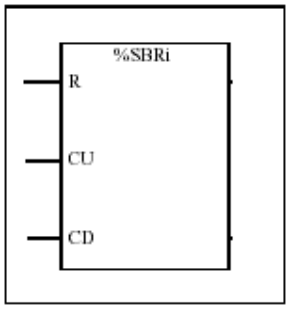
下面参数必须在配置中输入：

- 预置值(%Ci.P)：此例中设为5000
- 可调节：是

移位寄存器功能模块(%SBRi)

导言　　移位寄存器功能模块(%SBRi)提供了二进制数据位（0或1）的左移或右移。

图例　　下面是一个移位寄存器功能模块示例。

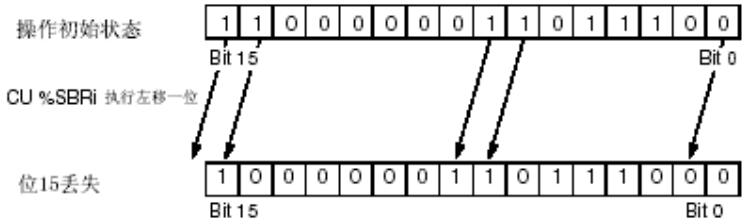


参数　　移位寄存器功能模块具有下列参数。

参数	标识	值
寄存器编号	%SBRi	0 到 7
寄存器位	%SBRi.j	移位寄存器的位0到15（j = 0到15）可被测试指令测试，且由赋值指令写。
输入（或指令）复位	R	其上升沿将寄存器位 0 到 15%SBRi.j 置为 0。
左移输入（或指令）	CU	其上升沿将寄存器位左移一位。
右移输入（或指令）	CD	其上升沿将寄存器位右移一位。

操作

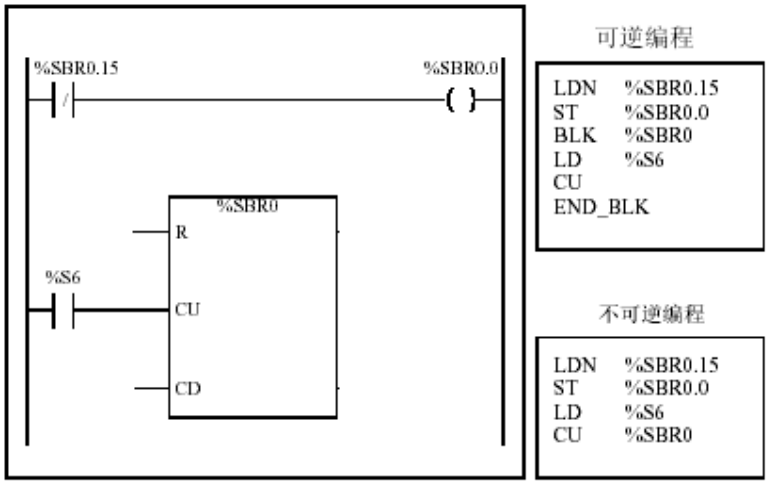
下图显示了移位操作前后的位形式。



使用指令CD右移一位（位15到位0）也是如此。位0被丢失。
如果一个16位寄存器不能满足要求，可以通过程序串连几个寄存器。

编程

下面示例中，每秒左移一位且假设位 0 与位 15 的状态相反。



特殊情况

下表包含了移位寄存器功能模块编程的特殊情况列表。

特殊情况	描述
冷重启(%S0=1)的影响	所有寄存器字的位被置为 0。
热重启(%S1=1)的影响	对寄存器字的位没有影响。

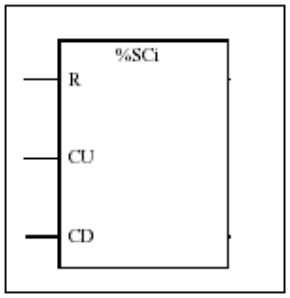
步进计数器功能模块(%SCi)

导言

步进计数器功能模块(%SCi)提供了一系列的步，动作可赋值给这些步。从一个步移动到另一个步取决于外部或内部事件。每当一个步处于激活状态时，相关位被置为 1。步进计数器在一个时刻只能有一个步被激活。

图例

下面是一个步进计数器功能模块示例。

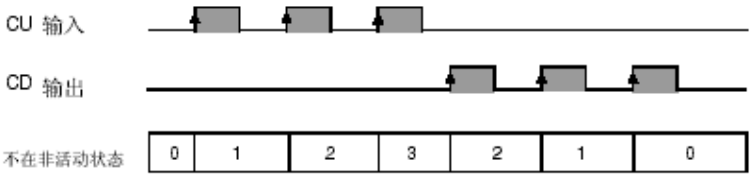


参数

步进计数器功能模块具有下列参数。

参数	标识	值
步进计数器编号	%SCi	0 到 7
步进计数器位	%SCi.j	步进计数器的位0到225（j = 0到225）可被装载逻辑测试，且由赋值指令写。
输入（或指令）复位	R	其上升沿将步进计数器复位。
输入（或指令）增加	CU	其上升沿将步进计数器增加一步。
输入（或指令）减少	CD	其上升沿将步进计数器减少一步。

时序图　　下面是步进计数器功能模块操作时序图。

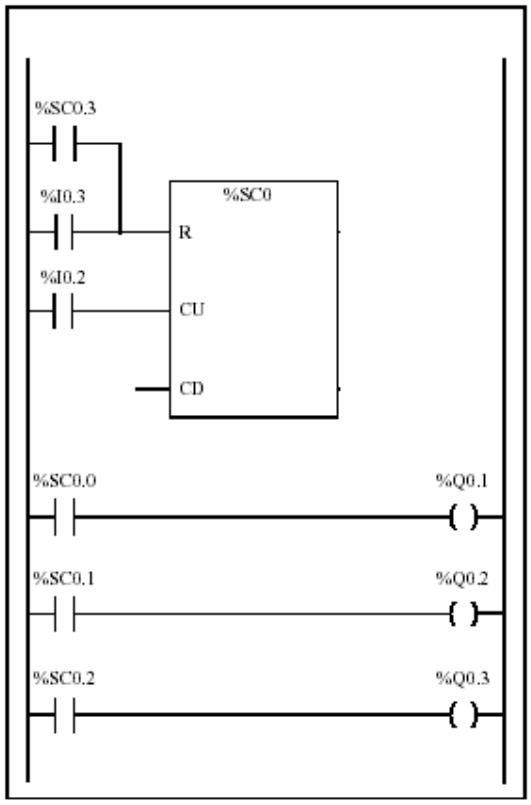


编程

下面是一个步进计数器功能模块示例。

- 步进计数器 0 由输入%I0.2 增加。
- 步进计数器 0 由输入%I0.3 或当它到达步 3 时复位到 0。
- 步 0 控制输出%Q0.1，步 1 控制输出%Q0.2，步 2 控制输出%Q0.3。

下图显示了此例的可逆和不可逆编程。



可逆编程

```
BLK     %SC0
LD      %SC0.3
OR      %I0.3
R
LD      %I0.2
CU
END_BLK
LD      %SC0.0
ST      %Q0.1
LD      %SC0.1
ST      %Q0.2
LD      %SC0.2
ST      %Q0.3
```

不可逆编程

```
LD      %SC0.3
OR      %I0.3
R       %SC0
LD      %I0.2
CU      %SC0
LD      %SC0.0
ST      %Q0.1
LD      %SC0.1
ST      %Q0.2
LD      %SC0.2
ST      %Q0.3
```

特殊情况

下表包含了步进计数器功能模块操作的特殊情况列表。

特殊情况	描述
冷重启(%S0=1)的影响	初始化步进计数器。
热重启(%S1=1)的影响	对步进计数器没有影响。

14.3 数字运算

概览

本节的主题　　本节提供了数字运算介绍，包括描述和编程指导。

本节包含了哪些内容？　　本节包含了以下主题：

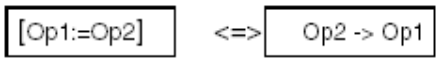
概览

数字指令一般用于 16 位字（见字对象，p.28）和 32 位双字（见浮点和双字对象，p.31）。它们写在方括号内。如果如果前面逻辑运算的结果为真（布尔运算累加器=1），则执行数字指令。如果如果前面逻辑运算的结果为假（布尔运算累加器=0），则不执行数字指令且操作数保持不变。

赋值指令

导言　　赋值指令用于把操作数 Op2 装入操作数 Op1。

赋值　　赋值指令语法。



赋值操作可用于：

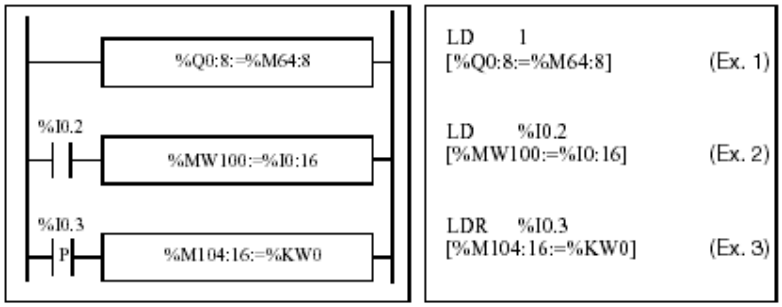
- 位串
- 字
- 双字
- 浮点字
- 字表
- 双字表
- 浮点字表

位串赋值　　该操作可用于下列位串（见结构对象，p.44）：

- 位串->位串（例 1）
- 位串->字（例 2）或双字（索引）
- 字或双字（索引）->位串（例 3）
- 立即值->位串

示例

位串赋值示例。



使用规则：

- 对位串->字赋值：位串中的位从右开始传送到字（位串的第一位送到字的第 0 位），并把字中没有被传送的位都置为 0（长度<=16）。
- 对字->位串赋值：字中的位从右开始传送（字的第 0 位被传送到位串的第一位）。

位串赋值

位串赋值的语法。

操作	语法	操作数 1（Op1）	操作数 2（Op2）
:=	[Op1: = Op2] 把操作数 2（Op2）的值赋给操作数 1（Op1）	%MWi,%QWi, %SWi %MWi[%MWi], %MDi, %MDi[%MWi] **%Mi:L, %Qi:L, %Si:L,** **%Xi:L**	立即值， %MWi, %KWi, %IW, %INWi, %QWi, %QNWi, %SWi, %BLK.x, %MWi[%MWi], %KWi[%MWi], %MDi[%MWi], %KDi[%MWi], **%Mi:L,%Qi:L, %Si:L,** **%Xi:L, %Ii:L**

注意：缩写%BLK.x（例如，%C0.P）用来表示任意功能模块字。

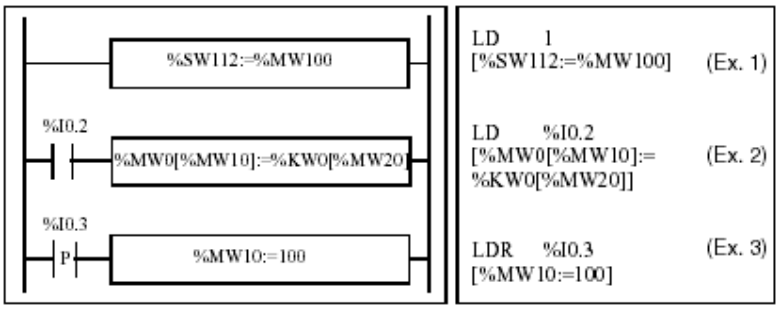
字赋值

赋值操作可用于下列字和双字：

- 字（索引）->字（例2）或双字（索引或非索引）
- 立即全值->字（例3）或双字（索引或非索引）
- 位串->字或双字
- 双字（索引或非索引）->双字或字（索引或非索引）
- 浮点（索引或非索引）->浮点（索引或非索引）
- 字或双字->位串
- 立即浮点值->浮点（索引或非索引）

示例

字赋值示例。



语法　　字赋值语法。

操作	语法
:=	[Op1: = Op2] 把操作数 2（Op2）的值赋给操作数 1（Op1）

下表给出了详细操作数：

类型	操作数 1（Op1）	操作数 2（Op2）
字， 双字， 位串	**%BLK.x, %MWi, %QWi, %SWi %MWi[MWi**, %MDi, %MDi[%MWj]], %Mi:L, %Qi:L, %Si:L, %Xi:L	**立即值, %MWi, %KWi, %IW, %QWi, %SWi, %MWi[MWi], %KWi[MWi], %MDi, %MDi[%MWj], %KDi, %KDi[MWj]** , %INW, %Mi:L, %Qi:L, %QNW, %Si:L, %Xi:L, %Ii:L
浮点	**%MFi, %MFi[%MWj]**	**立即浮点值, %MFi, %MFi[%MWj], %KFi, %KFi[%MWj]**

注意：缩写%BLK.x（例如，R3.I）用来表示任意功能模块字。对位串 %Mi:L, %Si:L, 和 %Xi:L，位串的第一个基地址必须是 8 的倍数(0, 8, 16, ..., 96, ...)。
字<-->双字转换在双字-->字赋值时时后台执行。如果双字的值不能包含在字中，则位%S18被置为1且双字的MSB丢失。

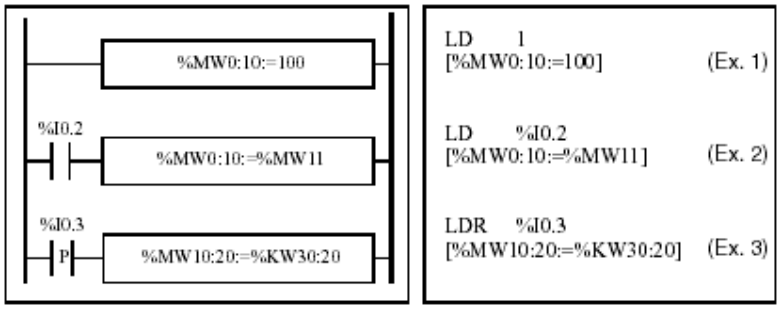
字，双字和浮点表赋值

赋值操作可用于下列对象表（见字表，p.45）：

- 立即全值->字表（例 1）或双字表
- 字->字表（例 2）
- 字表->字表（例 3）
  两个表的长度（L）应该相同。
- 双字->双字表
- 双字表->双字表
  两个表的长度（L）应该相同。
- 立即浮点值->浮点表
- 浮点->浮点表
- 浮点表->浮点表
  两个表的长度（L）应该相同。

示例

字表赋值示例。



语法　　字，双字和浮点表赋值语法

操作	语法
:=	[Op1: = Op2] 把操作数 2（Op2）的值赋给操作数 1（Op1）

下表给出了详细操作数：

类型	操作数 1（Op1）	操作数 2（Op2）
字表	**%MWi:L, %SWi:L**	**%MWi:L, %SWi:L**，立即全值, %MWi, %KWi, %IW, %QW, %SWi, %BLK.x
双字表	**%MDi:L**	**立即全值, %MDi,** **%KDi,%MDi:L, %KDi:L**
浮点表	**%MFi:L]**	**立即浮点值, %MFi,** **%KFi, %MFi:L, %KFi:L**

注意：缩写%BLK.x（例如，R3.I）用于描述任意功能模块字。

比较指令

导言

比较指令用于比较两个操作数。

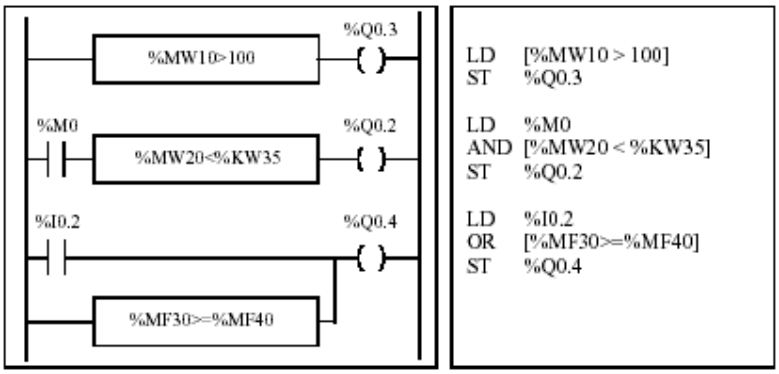
下表列出了比较指令类型。

指令	功能
>	测试操作数 1 是否大于操作数 2。
>=	测试操作数 1 是否大于或等于操作数 2。
<	测试操作数 1 是否小于操作数 2。
<=	测试操作数 1 是否小于或等于操作数 2。
=	测试操作数 1 是否等于操作数 2。
<>	测试操作数 1 是否不等于操作数 2。

结构

比较指令要加方括号，跟在指令LD, AND, 和 OR的后面。当被请求的比较为真时，结果为1。

比较指令示例。



语法

比较指令语法：

操作	语法
>, >=, <, <=, =, <>	LD [Op1 运算符 Op2] AND [Op1 运算符 Op2] OR [Op1 运算符 Op2]

操作数

类型	操作数1（Op1）	操作数2（Op2）
字	%MWi, %KWi, %INWi, %IW, %QNWi, %QWi, %QNWi, %SWi, %BLK.x	立即值, %MWi, %KWi, %INWi, %IW, %QNWi, %QW, %SWi, %BLK.x, %MWi [%MWi], %KWi [%MWi]
双字	%MDi, %KDi	立即值, %MDi, %KDi, %MDi [%MWi], %KWD [%MWi]
浮点字	%MFi, %KFi	立即浮点值, %MFi, %KFi, %MFi [%MWi], %KFi [%MWi]

注意：比较指令可用于圆括号内。

圆括号内比较指令使用示例：

```
LD      %M0
AND(    [%MF20 > 10.0]
OR      %I0.0
)
ST      %Q0.1
```

整数算术指令

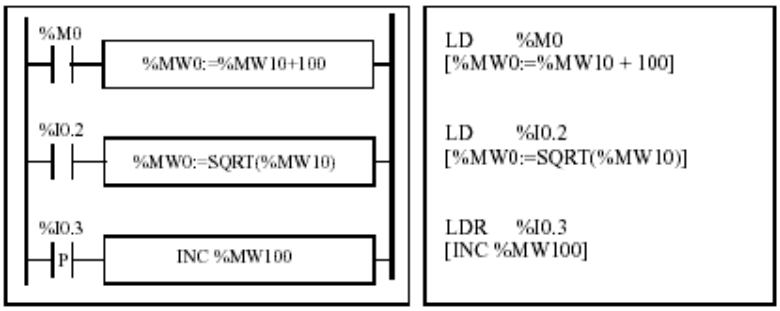
导言

算术指令用于执行两个整数操作数之间或一个整数操作数上的算术运算。下表列出了算术指令类型。

指令	功能
+	两个操作数相加
-	两个操作数相减
*	两个操作数相乘
/	两个操作数相除
REM	两个操作数相除的余数
SQRT	一个操作数的平方根
INC	一个操作数递增
DEC	一个操作数递减
ABS	一个操作数的绝对值

结构

算术运算如下所示：



语法

语法取决于使用何种运算符，如下表所示。

操作	语法
+,-,*,/,REM	[Op1: = Op 2 运算符 Op3]
INC, DEC	[运算符 Op1]
SQRT (1)	[Op1: = SQRT(Op2)]
ABS (1)	[Op1: = ABS(Op2)]

操作数：

类型	操作数 1（Op1）	操作数 2 和 3（Op2 & 3）（1）
字	%MWi, %QWi, %SWi	立即值, %MWi, %KWi, %INW, %IW, %QNW, %QW, %SWi, %BLK.x
双字	%MDi	立即值, %MDi, %KDi

注意：使用这个运算符时，Op2 不能是立即值。

溢出和出错条件

加法

- 字运算中溢出

如果结果超出-32768 和 +32767 范围，位%S18（溢出）被置为 1。且所得结果不正确（见下页例 1）。由用户程序管理位%S18。

- 绝对结果溢出（无符号算术运算）

在某些计算中，可能会用到将操作数用于无符号算术运算（这时第 15 位表示值 32768）。一个字操作数的最大值是 65535。两个绝对值（无符号）相加的和如果超过 65535 将导致溢出。这由系统位%S17（进位）变为 1 作为标志，表示值 65536。

注意：

对双字，其范围是-2147483648 和 21474836487。

减法

- 结果为负

如果减法的结果小于 0，系统位%S17 被置为 1。

乘法

- 运算时溢出

如果结果超出字的范围，位%S18（溢出）被置为 1，且结果没有意义。

除法/取余

- 被 0 除

如果除数是 0，则不能进行除法运算且系统位%S18 被置为 1。结果不正确。

- 运算时溢出

如果商超出字的范围，位%S18 被置为 1。

平方根开方

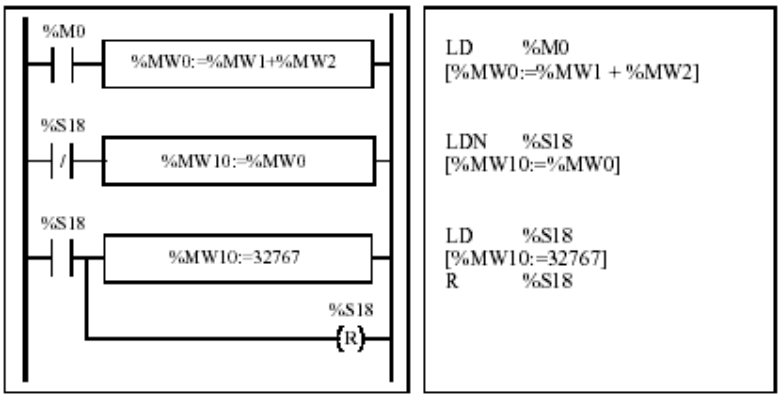
- 运算时溢出

平方根开方只适用正值。这样，其结果总为正。如果平方根操作数为负，系统位%S18 被置为 1 且结果不正确。

注意：用户程序用于管理系统位%S17 和%S18。它们被控制器置为 1，且必须由程序复位以便再次使用（见前面示例页）。

示例

示例 1：加法溢出。



如果%MW1 =23241 且 %MW2=21853，实际结果(45094)不能由一个 16 位字表示，位%S18 被置为 1 且获得的结果(-20442)不正确。此例中结果大于 32767 时，其值都固定为 32767。

示例2：[%MW2:=%MW0 + %MW1]其中%MW0 =65086, %MW1=65333字%MW2包含数字64883。位%S17被置为1且表现为值65536。无符号算术运算结果等于：65536 + 64883 =130419。

示例3：[%MW2:=%MW0 + %MW1] 其中 %MW0 =45736 (等于带符号值 -19800)，%MW1=38336 (t等于带符号值27200)。两个系统位%S17和 %S18被置为1。带符号算术运算结果(+18536)错误。无符号算术运算结果(18536 + %S17的值，等于84072)是正确的。

逻辑指令

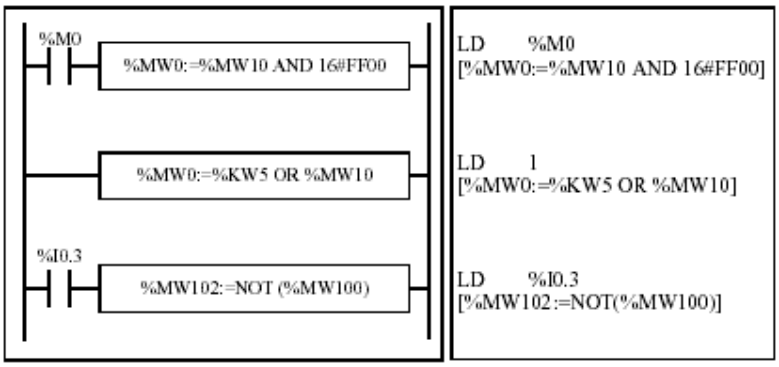
导言

逻辑指令用于执行两个字操作数之间或一个字操作数的逻辑运算。
下表列出了逻辑指令类型。

指令	功能
AND	与（位方式），用于两个操作数之间
OR	逻辑或（位方式），用于两个操作数之间
XOR	异或（位方式），用于两个操作数之间
NOT	逻辑反（位方式），用于一个操作数

结构

逻辑运算执行如下：



语法

语法取决于使用的运算符：

操作	语法	操作数 1（Op1）	操作数 2 和 3（Op2 & 3）
AND, OR, XOR	[Op1: = Op 2 运算符 Op3]	%MWi, %QWi, %SWi	立即值 (1), %MWi, %KWi, %IW, %QW, %SWi, %BLK.x
NOT	[NOT(Op2)]		

注意：(1) 在 NOT 指令中，Op2 不能是立即值。

示例

下面是一个逻辑与指令示例：

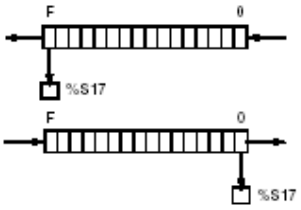
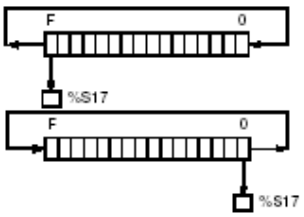
[%MW15 := %MW32 AND %MW12]

移位指令

导言

移位指令将操作数向右或向左移动若干位。

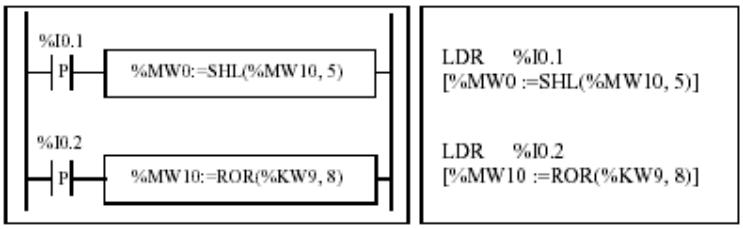
下表列出了移位指令类型。

指令	功能	
逻辑移位		
SHL(op2,i)	向左逻辑移动 i 位	
SHR(op2,i)	向右逻辑移动 i 位	
循环移位		
ROR(op2,i)	向左循环移动 i 位	
ROL(op2,i)	向右循环移动 i 位	

注意：系统位%S17（见系统位(%S)，p.454）用于容量超出状态。

结构

移位操作执行如下：



语法

语法取决于使用的运算符，如下表所示。

操作	语法
SHL, SHR	[Op1: = 运算符 (Op2,i)]
ROL, ROR	

操作数

类型	操作数 1（Op1）	操作数 2（Op2）
字	%MWi, %QWi, %SWi	%MWi, %KWi, %IW, %QW, %SWi, %BLK.x
双字	%MDi	%MDi, %KDi

转换指令

导言

转换指令执行数字不同类型间的转换。

下表列出了转换指令类型。

指令	功能
BTI	BCD --> 二进制 转换
ITB	二进制 --> BCD 转换

BCD 码介绍

Binary Coded Decimal (BCD)用四位二进制码表示表示一个十进制数（0 到 9）。这样一个 16 位字对象可以用四个十进制数字表示（0000 - 9999），一个 32 位双字对象可以包含 8 个十进制数字。

转换中，如果值不是BCD码，则位%S18置为1。该位必须由程序测试和复位到0。

十进制数的BCD码表示：

十进制	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

示例：

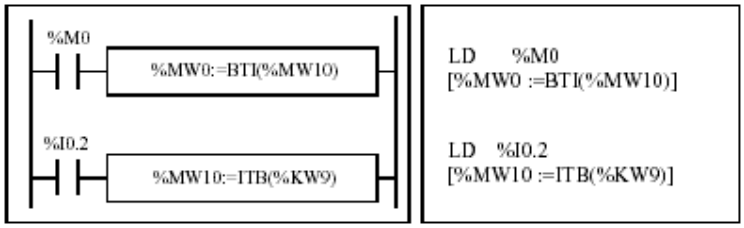
- 字%MW5 的 BCD 值"2450"对应二进制值：0010 0100 0101 0000
- 字%MW12 的十进制值"2450"对应二进制值：0000 1001 1001 0010

用指令 BTI 将字%MW5 转换成字%MW12。

用指令 ITB 将字%MW12 转换成字%MW5。

结构

转换操作执行如下：



语法

语法取决于使用的运算符，如下表所示。

操作	语法
BTI, ITB	[Op1: = 运算符 (Op2,i)]

操作数

类型	操作数 1（Op1）	操作数 2（Op2）
字	%MWi, %QWi, %SWi	%MWi, %KWi, %IW, %QW, %SWi, %BLK.x
双字	%MDi	%MDi, %KDi

应用示例

BTI 指令用来处理拨码开关的 BCD 码在控制器输入的设定值。
ITB 指令用来以 BCD 码显示数字值（如，计算结果，功能模块的当前值）。

单/双字转换指令

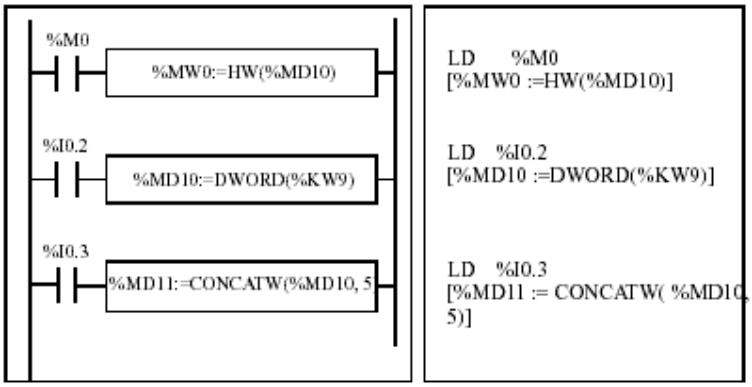
导言

下表描述了用于执行单字和双字之间转换的指令：

指令	功能
LW	抽取双字的 LSB 到一个字。
HW	抽取双字的 MSB 到一个字。
CONCATW	连接两个字到一个双字。
DWORD	转换一个 16 位字到一个 32 位字。

结构

转换操作执行如下：



语法

语法取决于使用的运算符，如下表所示：

操作	语法	操作数 1 (Op1)	操作数 2 (Op2)	操作数 3 (Op3)
LW, HW	Op1= 运算符 (Op2)	%MWi	%MDi, %KDi	[-]
CONCATW	Op1 = 运算符 (Op2, Op3))	%MDi	%MWi, %KWi, 立即值	%MWi, %KDi, 立即值
DWORD	Op1 = 运算符 (Op2)	%MDi	%MWi, %KWi	[-]

14.4 程序指令

概览

本节的主题　　本节提供了程序指令介绍。

本节包含了哪些内容?　　本节包含了以下主题:

导言

END 指令定义一个程序扫描执行的结束。

END, ENDC, 和 ENDCN

有三个不同的结束指令可用：

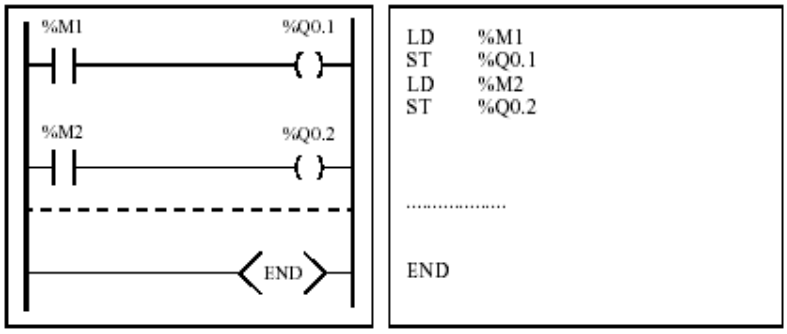
- END：程序无条件结束
- ENDC：如果前面测试指令布尔运算结果是 1，则程序结束。
- ENDCN：如果前面测试指令布尔运算结果是 0，则程序结束。

默认（正常模式）情况下，当程序结束被激活时，输出被更新且开始下一次扫描。

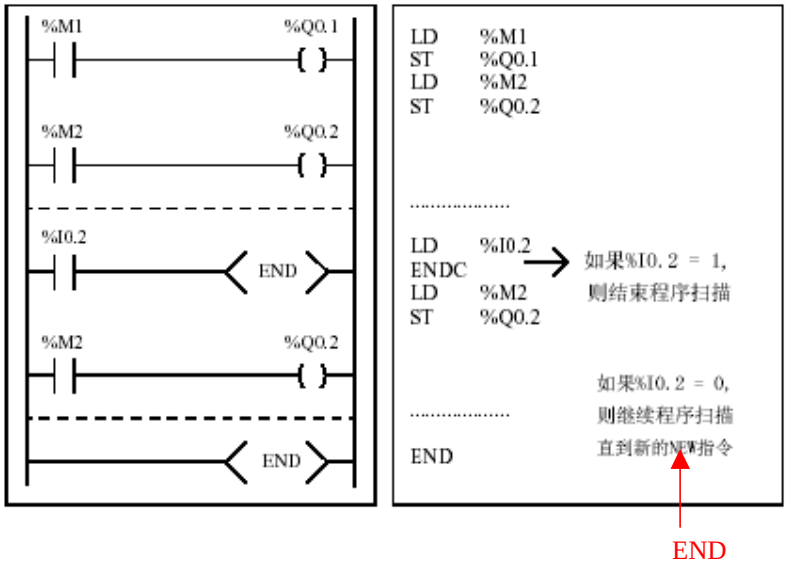
如果是周期扫描，则当周期结束时输出被更新且开始下一次扫描。

示例

无条件 END 指令示例。



有条件 END 指令示例。



NOP 指令

NOP	NOP 指令不执行任何操作。用它在程序中“保留”行，以便您以后插入指令而无需修改行号。

跳转指令

导言

跳转指令使程序执行立即中断并转入执行标号为%Li（i = 1 到 16 对于一体型控制器，1 到 63 对于其它控制器）的程序行。

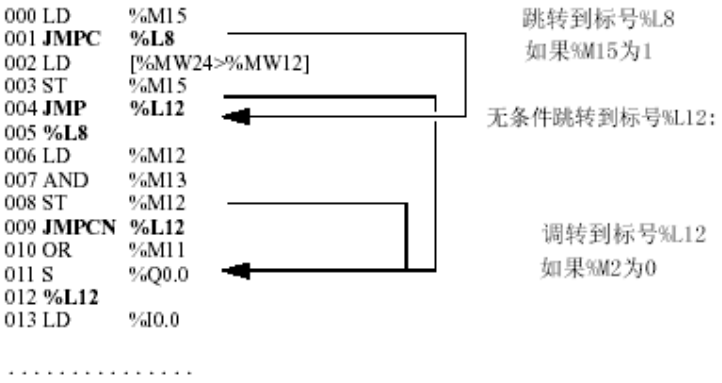
JMP, JMPC 和 JMPCN

有三个不同的跳转指令可用：

- JMP：程序无条件跳转
- JMPC：如果前面逻辑布尔运算结果为 1，则程序跳转
- JMPCN：如果前面逻辑布尔运算结果为 0，则程序跳转

示例

跳转指令示例。



指导

- 跳转指令不允许用于圆括号内，且不能位于指令 AND(, OR(和右括号指令")"之间。
- 标号只能位于指令 LD, LDN, LDR, LDF 或 BLK 之前。
- 标号%Li 的编号在程序中只能被定义一次。
- 程序可以向下或向上跳转。当向上跳转时，必须注意程序扫描时间。延长扫描时间可能导致看门狗的触发。

导言　　子程序指令导致程序执行一个子程序，然后返回到主程序。

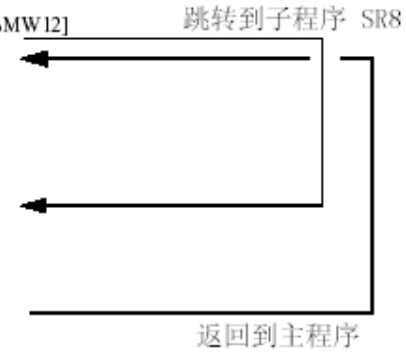
SRn, SRn: 和 RET

子程序由三步组成：

- SRn 指令调用标号为 SRn 的子程序，如果前面布尔指令结果为 1。
- 子程序用标号 **SRn:**表示，n = 0 到 15 对于 TWDLCAA10DRF, TWDLCAA16DRF，0 到 63 对于其它控制器。
- **RET** 指令位于子程序的最后，返回到主程序。

示例　　子程序指令示例。

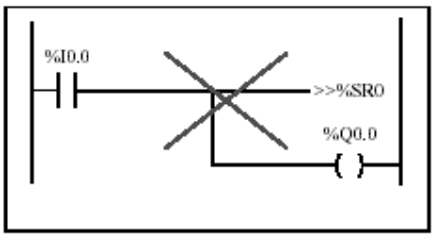
```
000  LD     %M15
001  AND    %M5
002  ST     %Q0.0
003  LD     [%MW24>%MW12]
004  SR8
005  LD     %I0.4
006  AND    M13
007  _
008  _
009  _
010  END

011  SR8:
012  LD     1
013  IN     %TM0
014  LD     [%TM0.Q]
015  ST     %M15
010  RET
```

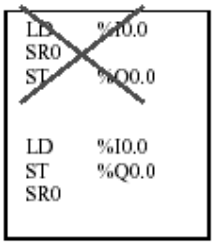


指导方针

- 一个子程序不能调用另一个子程序。
- 子程序指令不允许用于圆括号内，且不能位于指令 AND(, OR(和右括号指令")"之间。
- 标号只能位于指令 LD 或 BLK 之前，用于标识一个布尔等式（或梯级）的开始。
- 赋值指令不能跟随在子程序调用之后。这是因为子程序可能改变布尔运算累加器的内容。这样返回时，它的值可能与调用前不同。见下面示例。

子程序编程示例。





15

概览

本章的主题

本章提供了有关用于创建 Twido 可编程控制器高级控制程序的指令和功能模块的详细资料。

本章包含了哪些内容?

本章包含了以下几节:

15.1 高级功能模块

概览

本节的主题　　本节提供了高级功能模块介绍，包括编程示例。

本节包含了哪些内容？　本节包含了以下主题：

与高级功能模块有关的位和字对象

导言

高级功能模块使用与标准功能模块类似的专用字和位类型。高级功能模块包括：

- LIFO/FIFO 寄存器(%R)
- 鼓型控制器(%DR)
- 高速计数器(%FC)
- 超高速计数器(%VFC)
- 脉宽调制输出(%PWM)
- 脉冲发生器输出(%PLS)
- 移位寄存器(%SBR)
- 移位计数器(%SC)
- 消息控制模块(%MSG)

程序可访问的对象

下表包含了与各种高级功能模块相关的程序可访问字和位的概览。请注意下表中的写访问依赖于配置中选择的“可调节”设置。通过 TwidoSoft 或操作接口设置允许或拒绝对字或位的访问。

高级功能模块	相关字和位		地址	写访问
%R	字	寄存器输入	%Ri.I	可以
	字	寄存器输出	%Ri.O	可以
	位	寄存器输出满	%Ri.F	不可以
	位	寄存器输出空	%Ri.E	不可以
%DR	字	当前步号	%DRi.S	可以
	位	当前步为最后一步	%DRi.F	不可以
%FC	字	当前值	%FCi.V	不可以
	字	预置值	%FCi.P	可以
	位	完成输出	%FCi.D	不可以
%VFC	字	当前值	%VFCi.V	不可以
	字	预置值	%VFCi.P	可以
	位	计数方向	%VFCi.U	不可以
	字	捕获值	%VFCi.C	不可以
	字	阈值 0	%VFCi.SO	可以
	字	阈值 1	%VFCi.S1	可以
	位	溢出	%VFCi.F	不可以
	位	计算的频率	%VFCi.M	可以
	位	映像输出 0 使能	%VFCi.R	可以
	位	映像输出 1 使能	%VFCi.S	可以
	位	阈值输出 0	%VFCi.TH0	不可以
	位	阈值输出 1	%VFCi.TH1	不可以
	位	频率测量时基	%VFCi.T	可以
%PWM	字	占空比	%PWMi.R	可以
	字	预置周期	%PWMi.P	可以
%PLS	字	脉冲数	%PLSi.N	可以
	字	预置值	%PLSi.P	可以
	位	当前输出使能	%PLSi.Q	不可以
	位	发生完成	%PLSi.D	不可以

高级功能模块	相关字和位		地址	写访问
%SBR	位	寄存器位	%SBRi.J	可以
%SC	位	步进计数器位	%SCi.j	可以
%MSG	位	完成	%MSGi.D	不可以
	位	出错	%MSGi.E	不可以

概览

所有的 Twido 应用程序以列表程序的格式存储，即使在梯形图编辑器中写成。因此，Twido 控制器又被称为列表“机器”。术语“可逆性”指 TwidoSoft 具有能力：将列表程序用梯形图表示，反之亦然。默认情况下，所有梯形图程序都是可逆的。

和基本功能模块一样，高级功能模块也必须考虑可逆性规则。列表语言中可逆功能模块的结构需要用到下列指令：

- **BLK**：标志模块的开始和功能模块的输入部分
- **OUT_BLK**：标志功能模块输出部分的开始
- **END_BLK**：标志功能模块的结束

注意：这些可逆功能模块指令的使用对一个正确功能的列表程序来说并不是强制性的。一些指令可用于列表语言编程但并不可逆。

专用输入和输出

高速计数器，超高速计数器，PLS，和 PWM 高级功能使用专用输入和输出，但这些位不被任何一个模块专用所保留。而且，当使用这些高级功能时必须管理这些专用资源的使用，您必须管理怎样分配专用输入和输出。TwidoSoft 帮助配置这些资源，显示输入/输出配置详细信息，并在一个专用输入或输出已被配置的功能模块使用时提出警告。

下表是专用输入和输出及其特殊功能概括。

当用于计数功能时：

输入	使用
%I0.0.0	%VFC0: 加/减管理或相位B
%I0.0.1	%VFC0: 脉冲输入或相位A
%I0.0.2	%FC0: 脉冲输入或%VFC0预置输入
%I0.0.3	%FC1: 脉冲输入或%VFC0捕获输入
%I0.0.4	%FC2: 脉冲输入或%VFC1捕获输入
%I0.0.5	%VFC1 预置输入
%I0.0.6	%VFC1: 加/减管理或相位B
%I0.0.7	%VFC1: 脉冲输入或相位A

当用于计数或特殊功能时：

输出	使用
%Q0.0.0	%PLS0 或 PWM0 输出
%Q0.0.1	%PLS1 或 PWM1 输出
%Q0.0.2	%VFC0 映像输出
%Q0.0.3	
%Q0.0.4	%VFC1 映像输出
%Q0.0.5	

专用输入和输出使用

TwidoSoft 对专用输入和输出的使用规定了下面规则。

- 每个使用专用 I/O 的功能模块必须被配置，然后被应用程序引用。专用 I/O 只能在配置功能模块时被分配，程序引用时不能分配。
- 功能模块被配置后，它的专用输入和输出不能被应用程序或另一功能模块使用。

  例如，如果您配置%PLS0，您不能在%DR0（磁鼓控制器）或应用程序逻辑（如 ST %Q0.0.0）中使用%Q0.0.0。
- 如果一个功能模块所需的专用输入或输出已被应用程序或另一功能模块使用，则这个功能模块不能被配置。

  例如，如果您配置%FC0 作为一个加计数器，则您不能配置%VFC0 使用%I0.0.2 作为捕获输入。

注意：为了改变专用 I/O 的使用，可以通过设置对象的类型到“未使用”来取消功能模块配置，然后移动引用到应用程序中的功能模块。

LIFO/FIFO 寄存器功能模块(%Ri)

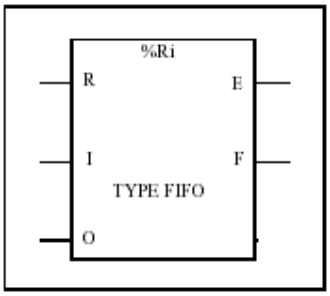
导言

寄存器是存储器模块，可以存储最多 16 个 16 位的字。有两种不同的存储方式：

- 队列方式（先入先出）即 FIFO。
- 堆栈方式（后入先出）即 LIFO。

图例

下面是一个寄存器功能模块图例。



寄存器功能模块

参数

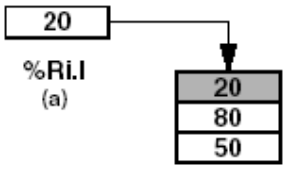
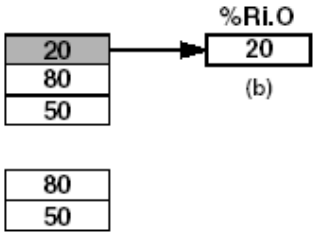
寄存器功能模块具有如下参数：

参数	标识	值
寄存器编号	%Ri	0 到 3。
类型	FIFO或 LIFO	队列方式或堆栈方式。
输入字	%Ri.I	寄存器输入字，可读取，测试，和写入。
输出字	%Ri.O	寄存器输出字，可读取，测试，和写入。
存储输入 （或指令）	I (In)	上升沿时将字%Ri.I 的内容存入寄存器。
取回输入 （或指令）	O (Out)	上升沿时将一个数据字装入字%Ri.O 内。
复位输入 （或指令）	R (Reset)	状态为 1 时，初始化寄存器。
空输出	E (Empty)	对应位%Ri.E 表示寄存器空。可测试。
满输出	F (Full)	对应位表%Ri.F 示寄存器满。可测试。

LIFO 操作

导言　　LIFO 操作（后入先出）中，最后进入的数据项最先被取出。

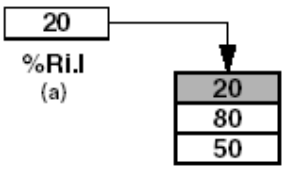
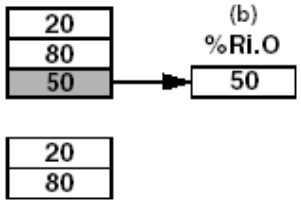
操作　　下表是 LIFO 操作描述。

步骤	描述	示例
1	当接收到一个存储请求（输入 I 的上升沿或指令 I 被激活），输入字%Ri.I 的内容（已经被装入）被存放到堆栈的顶部（图 a）。当堆栈已满（输出=1）时，不能再存入数据。	存储%Ri.I 的内容到堆栈的顶部。 
2	当接收到一个取回请求（输入 O 的上升沿或指令 O 被激活），堆栈顶部的数据字（最后进入的字）被装入字%Ri. O（图 b）。当寄存器为空（输出 E=1）时，不能再取出数据。输出字%Ri.O 不变且保持原值。	取出堆栈中最上面数据字。 
3	堆栈可在任何时候被复位（输入 R 状态为 1 或指令 R 被激活）。指针指向堆栈的顶端元素。	

FIFO 操作

导言　　FIFO 操作（先入先出）中，最先进入的数据项最先被取出。

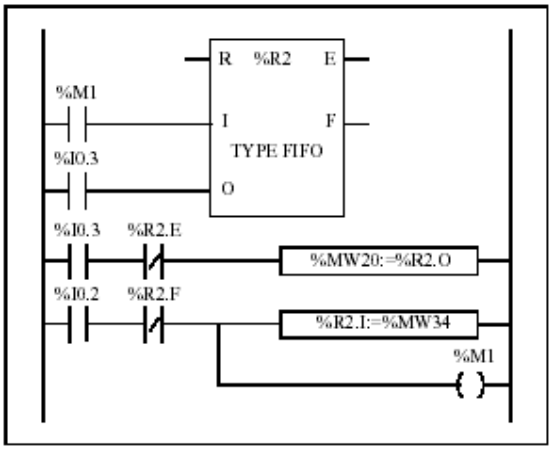
操作　　下表是 FIFO 操作描述。

步骤	描述	示例
1	当接收到一个存储请求（输入 I 的上升沿或指令 I 被激活），输入字%Ri.I 的内容（已经被装入）被存放到队列的顶部（图 a）。当队列已满（输出=1）时，不能再存入数据。	存储%Ri.I 的内容到队列的顶部。 
2	当接收到一个取回请求（输入 O 的上升沿或指令 O 被激活），队列最底部的数据字被装入字%Ri. O，且寄存器队列中的数据字都往底部移动一格（图 b）。当寄存器为空（输出 E=1）时，不能再取出数据。输出字%Ri.O 不变且保持原值。	取出第一个数据项装入%Ri.O 中。 
3	队列可在任何时候被复位（输入 R 状态为 1 或指令 R 被激活）。	

导言

下面编程示例显示了寄存器%R2 未满(%R2.F = 0)时，使用存储请求(%I0.2)将一个存储字(%MW34)的内容装入寄存器(%R2.I)。寄存器的存储请求由%M1 实现。取出请求由输入%I0.3 来实现，且如果寄存器不为空（%R2.E = 0），则%R2.O 的内容被装入%MW20。

编程示例　　下图是一个寄存器功能模块可逆和不可逆编程示例。



梯形图

```
BLK          %R2
LD           %M1
I
LD           %I0.3
O
END_BLK
LD           %I0.3
ANDN         %R2.E
[%MW20:=%R2.O]
LD           %I0.2
ANDN         %R2.F
[%R2.I:=%MW34]
ST           %M1
```

可逆编程

```
LD           %M1
I            %R2
LD           %I0.3
O            %R2
ANDN         %R2.E
[%MW20:=%R2.O]
LD           %I0.2
ANDN         %R2.F
[%R2.I:=%MW34]
ST           %M1
```

不可逆编程

配置

配置时必须输入寄存器类型这个参数：

- FIFO（默认），或
- LIFO

特殊情况

下表列出了寄存器功能模块编程的特殊情况：

特殊情况	描述
冷重启(%S0=1)的影响	初始化寄存器的内容。与输出 E 相关的输出位%Ri.E 被置为 1。
控制器停止后热重启(%S1=1)的影响	对寄存器的当前值和输出位的状态没有影响。

脉宽调制功能模块(%PWM)

导言

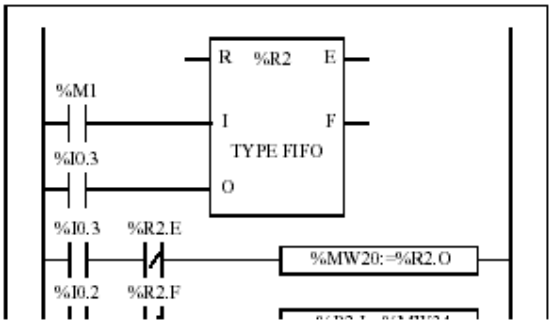
脉宽调制(%PWM)功能模块在专用输出通道%Q0.0.0 或 %Q0.0.1 生成一个方波信号，脉宽可变，因此占空比也可变。由于频率限制，带有继电器输出的控制器的这两个通道不支持这个功能。

有两个%PWM模块可用。%PWM0使用专用输出%Q0.0.0，%PMW1使用专用输出%Q0.0.1。%PLS功能模块也使用这两个输出，因此您必须在这两个功能之间做出选择。

图例

PWM 模块和时序图：

这张图是错的，是寄存器功能块的图！！！



参数

下表列出了 PWM 功能模块参数。

参数	标识	描述
时基	TB	0.142 ms, 0.57 ms, 10 ms, 1 s (默认值)
预置周期	%PWMi.P	1 < %PWMi.P <= 32767 如果时基为10 ms或1 s 0 < %PWMi.P <= 255 如果时基为0.57 ms或0.142 s 0 = 功能未使用 为了占空比获得好的精度级，推荐使用%PWMi.P >= 100
占空比	%PWMi.R	该值表示状态为 1 的信号在周期中的百分比。宽度 Tp 等于：Tp = T * (%PWMi.R/100)。用户应用程序写%PWMi.R 值。这个字控制周期的占空比。对于 T 定义，见下面“周期范围”。默认值为 0，当值大于 100 时视为等于 100。
脉冲发生输入	IN	状态为 1 时，脉宽调制信号在输出通道发生。状态为 0 时，输出通道被置为 0。

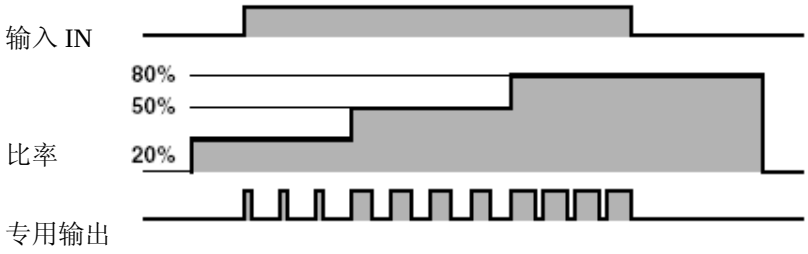
周期范围

预置值和时基可在配置中修改。它们用于确定信号周期 T=%PWMi.P * TB。获得的比率越小，则选择的%PWMi.P 越大。可用的周期范围：

- 时基 0.142 ms 时，0.142 ms 到 36.5 ms (27.4Hz 到 7kHz)
- 时基 0.57 ms 时，0.57 ms 到 146 ms (27.4Hz 到 7kHz)
- 时基 10 ms 时，10 ms 到 5.45 mins
- 时基 1 sec 时，1 sec 到 9.1 hours

操作

输出信号的频率在配置时通过选择时基 TB 和预置%PWMi.P 来设置。在程序中修改占空比% PWMi.R 来调制信号的宽度。下面是 PWM 功能模块占空比改变时的脉冲图示。



编程和配置

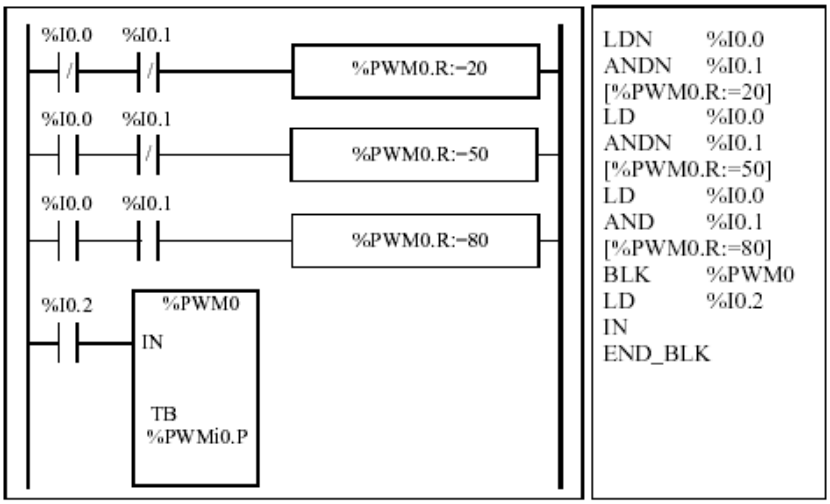
在这个示例中，信号宽度由程序根据控制器输入%I0.0.0 和 %I0.0.1 的状态来修改。

如果%I0.0.1 和 %I0.0.2置为0，%PWM0.R比率置为20%，则信号在状态1的持续时间为：20 % x 500 ms = 100 ms。

如果%I0.0.0置为0，%I0.0.1置为1，%PWM0.R比率置为50%（持续时间为250 ms）。

如果%I0.0.0 和 %I0.0.1置为1，%PWM0.R比率置为80%（持续时间为400 ms）。

编程示例：



特殊情况

下表显示了 PWM 功能模块特殊操作列表。

特殊情况	描述
冷重启(%S0=1)的影响	设置%PWMi.R 比率为 0。另外，%PWMi.P 复位到配置预置值，且这将取代活动表编辑器或可选操作显示中完成的任何变化。
热重启(%S1=1)的影响	没有影响。
输出专用给%PWM 模块的影响	使用一个编程设备强制输出%Q0.0.0 或%Q0.0.1 不会停止信号的生成。

脉冲发生器输出功能模块（%PLS）

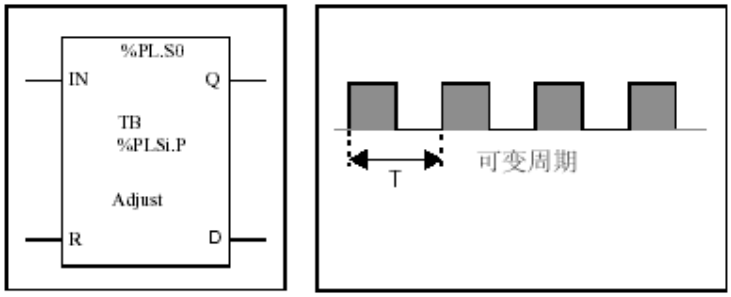
导言

%PLS 功能模块用于生成方波信号。有两个%PLS 功能模块可用于专用输出通道%Q0.0.0 或 %Q0.0.1。%PLS 功能模块只允许一个信号宽度，既 50%的占空比。当脉冲序列执行时，您可以选择限制脉冲的数目或周期。这可由配置的时间决定，和/或被用户应用程序更新。

注意：若控制器的这两个通道为继电器输出，则不支持%PLS 功能。

表示

脉冲发生器功能模块示例：



- TON=T/2 对于 0.142ms 和 0.57ms 时基
  = (%PLi.P*TB)/2
- TON=[所有部分(%PLi.P)/2]*TB 对于 10ms 到 1s time 时基。

说明

下表为 PLS 功能模块特性：

功能	对象	描述
时基	TB	0.142 ms, 0.57 ms, 10 ms, 1 sec
预置周期	%PLSi.P	当达到%PLS1.N 时%PLS1 脉冲输出不会停止。 它只对%PLS0 有效。 1 < %PLSi.P <= 32767 对于时基 10ms 或 1sec; 0 < %PLSi.P<= 255 对于时基 0.57 ms 或 0.142 ms; 0 = 功能未被使用。
脉冲数目	%PLSi.N	周期 T 内生成的脉冲数目，被限制在 0 < %PLSi.N < 32767。默认值置为 0。若要生成无限数目的脉冲，将%PLSi.N 置为 0。不论调节的设置情况如何，脉冲数目可以被改变。
可调节	Y/N	如果置为 Y，则可以通过 HMI 或动态数据编辑器修改预置值%PLSi.P。如果置为 N 则不能访问这个预置值。
脉冲发生器输入	IN	状态为 1 时，脉冲生成于专用输出通道。状态为 0 时，输出通道被置为 0。
复位输入	R	状态为 1 时，输出%PLSi.Q 和 %PLSi.D 被置为 0。周期 T 内生成的脉冲数目被置为 0。
当前脉冲生成输出	%PLSi.Q	状态为 1 时，表示脉冲信号在配置的专用输出通道生成。
脉冲生成完成输出	%PLSi.D	状态为 1 时，信号生成完成。达到期望的脉冲数目。

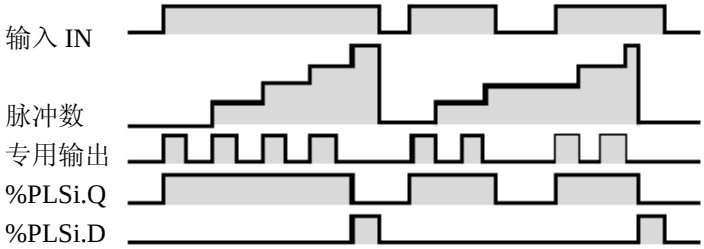
周期范围

预置值和时基可在配置中修改。它们用于确定信号周期 T=%PWMi.P * TB。可用的周期范围：

- 时基 0.142 ms 时，0.142 ms 到 36.5 ms (27.4Hz 到 7kHz)
- 时基 0.57 ms 时，0.57 ms 到 146 ms (6.84 Hz 到 1.75 kHz)
- 时基 10 ms 时，20 ms 到 5.45 mins
- 时基 1 sec 时，2sec 到 9.1 hours

操作

下面是%PLS 功能模块图例。



特殊情况

特殊情况	描述
冷重启(%S0=1)的影响	设置%PLSi.P 到配置定义值
热重启(%S1=1)的影响	没有影响
修改预置值(%PLSi.P)的影响	立即生效
使用用于%PLS 模块的输出的影响	使用一个编程设备强制输出%Q0.0.0 或%Q0.0.1 不会停止信号的生成。

注意：%PLSx.D 当达到期望的脉冲数目时被置位。可以设置输入 IN 或 R 到 1 将其复位。

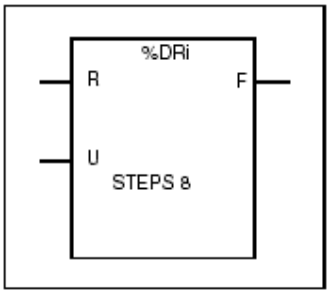
鼓型控制器功能模块(%DR)

导言

鼓型控制器的工作原理与机电类凸轮控制器类似，即根据外部事件改变步序。在每一步，凸轮的高点给出一个命令让控制器执行。鼓型控制器中，每一步的高点用状态 1 来表示，且作为控制位被赋给输出位%Qi.j 或内部位%Mi。

图例

下面是鼓型控制器功能模块图例。



鼓型控制器功能模块

参数

鼓型控制器功能模块具有如下参数。

参数	标识	值
编号	%DRi	0 到 3 对于一体型控制器，0 到 7 对于模块型控制器
当前步号	%DRi.S	0<%DRi.S<7。该字可被读和写。被写值必须是十进制立即值。被写后，在功能模块下次执行时生效。
步数		1 到 8（默认值）
回到 0 步输入（或指令）	R (Reset)	状态为 1 时，将鼓型控制器置为步 0。
前进输入（或指令）	U (Upper)	上升沿使鼓型控制器前进一步并更新控制位。
输出	F (Full)	表示当前步等于定义的最后一步。相关位可被测试（例如，如果%DRi.S=配置的步数-1，则%DRi.F=1）。
控制位		与步（16 位控制位）相关的输出或内部位，在配置编辑器中被定义。

导言

鼓型控制器包括：

- 由 8 步和 16 个数据位（步的状态）组成的常数（凸轮）矩阵，数据列编号为 0 到 F。
- 一个控制位列表，它与配置输出(%Qi.j.k)或存储字(%Mi)相对应。在当前步，控制位为该步定义的二进制状态。

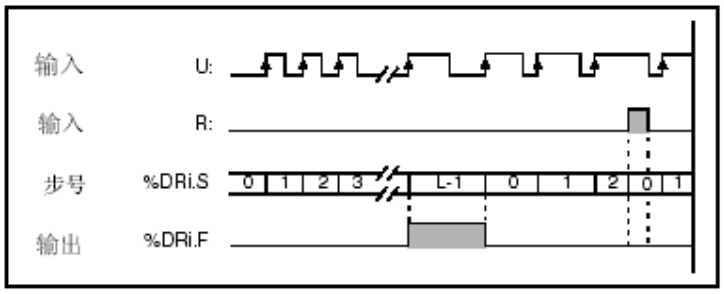
下表中的示例概括了磁鼓控制器的主要特性。

操作

上面示例中，步5是当前步，控制位%Q0.1, %Q0.3, 和%Q1.5被置为状态1；控制位%Q0.6, %Q0.5, 和 %Q1.0被置为状态0。当前步号在输入U的每个上升沿处（或指令U被激活时）增加。当前步可通过程序修改。

时序图

下图是鼓型控制器操作说明。



特殊情况

下表包含了鼓型控制器操作的特殊情况列表。

特殊情况	描述
冷重启(%S0=1)的影响	复位鼓型控制器到步 0（控制位更新）。
热重启(%S1=1)的影响	当前步后更新控制位。
程序跳转的影响	磁鼓控制器不再被扫描，意味着控制位不被复位。
控制位更新	只在步发生变化或冷热重启时发生。
主控制器继电器指令 MCS/MCR 的影响	两个 MCS/MCR 指令间有一个磁鼓控制器意味着如果指令 MCS 前面的指令布尔运算结果为 0 的话，则控制位被复位到 0。

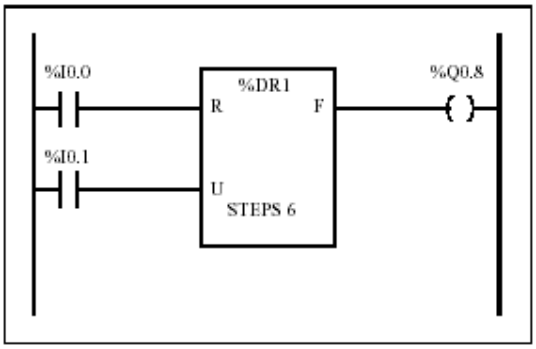
鼓型控制器编程和配置

导言

下面是一个鼓型控制器编程和配置示例。每次输入%I0.1 被置为 1，则前面六个输出%Q0.0 到 %Q0.5 被激活。输入 I0.0 将输出复位到 0。

编程示例

下面是一个磁鼓控制器可逆和不可逆编程示例。



梯形图

```
BLK     %DR1
LD      %I0.0
R
LD      %I0.1
U
OUT_BLK
LD      F
ST      %Q0.8
END_BLK
```

可逆编程

```
LD      %I0.0
R       %DR1
LD      %I0.1
U       %DR1
LD      %DR1.F
ST      %Q0.8
```

不可逆编程

配置

下面信息在配置中定义：

- 步数：6
- 鼓型控制器每步的输出状态（控制位）。

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
步 1:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
步 2:	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
步 3:	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
步 4:	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
步 5:	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
步 6:	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

- 控制位分配。

1 :	%Q0.0	4 :	%Q0.1
2 :	%Q0.2	5 :	%Q0.3
3 :	%Q0.4	6 :	%Q0.5

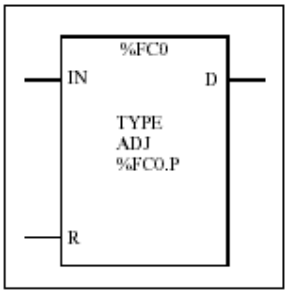
高速计数器功能模块

导言

高速计数器功能模块(%FC)可用作加计数器或减计数器。它能对数字输入的上升沿进行计数，频率最高 5kHz。因为高速计数器由特殊硬件中断管理，所以保持的最高频率采样率可根据您的特殊应用和硬件配置来改变。一体型控制器可最多配置使用三个高速计数器，而模块型控制器最多只能使用两个高速计数器。高速计数器功能模块%FC0, %FC1, 和%FC2分别使用专用输入%I0.0.2, %I0.0.3, 和%I0.0.4。这些位不保留为专用。其它功能模块使用这些专用资源时必须考虑它们的分配。

图例

下面是一个高速计数器功能模块示例。



参数

下表列出了高速计数器功能模块的参数。

参数	标识	描述
功能	TYPE	配置中设置，可设置为加计数或减计数。
预置值	%FCi.P	初始值设置，在 1 和 65535 之间。
可调节	Y/N	如果设为Y，则可以使用操作显示或动态数据表编辑器修改预置值%FCi.P 和 %FCi.V。如果设为N，则不能访问预置值。
当前值	%FCi.V	当前值根据选择的加或减计数功能增加或减少。对加计数，当前值复位到 0 且递增计数直至 65536。对减计数，当前值复位到预置值%FCi.P，且递减计数直至 0。
输入使能	IN	状态为 1 时，当前值根据应用到物理输入的脉冲更新。状态为 0 时，当前值保持上次值不变。
复位	%FCi.R	用于初始化模块。状态为 1 时，如果配置为加计数器则将当前值复位到 0，如果配置为减计数器则当前值被置为%FCi.P。完成位%FCi.D 被置回到它的默认值。
完成	%FCi.D	当配置为加计数器且%FCi.V 到达%FCi.P，或配置为减计数器且%FCi.V 到达 0 时，该位被置为 1。该位只读，只能通过置%FCi.R 为 1 将其复位。

特别注意

如果配置为可调节，则应用程序可以在任何时间改变预置值%FCi.P 和当前值%FCi.V。但是，新值仅在输入复位激活时或在输出%FCi.D 的上升沿生效。这使得不同的计数得到延续而不丢失脉冲。

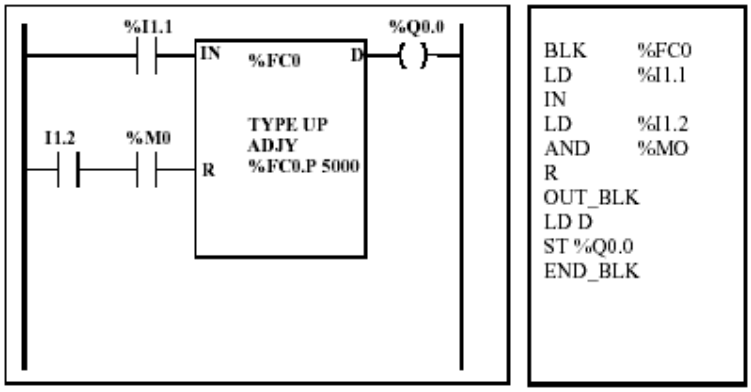
操作

如果配置为加计数，当专用输入得到一个上升沿时，当前值增加 1。当到达预置值%FCi.P 时，完成输出位%FCi.D 被置为 1 且当前值%FCi.V 被装入 0。

如果配置为减计数，当专用输入出项一个上升沿时，当前值增加1。当值为0时，完成输出位%FCi.D被置为1且预置值被装入到当前值%FCi.V。

配置和编程

此例中，当%I1.1被置为1时，应用程序对高达5000条的项目进行计数。%FC0的输入是专用输入%I0.0.2。当到达预置值时，%FC0.D被置为1且保持直至%FC0.R 得到由%I1.2和%M0"逻辑与"的结果给出的命令时复位。



特殊情况

下表包含了%FC 功能模块的特殊操作情况列表：

特殊情况	描述
冷重启(%S0=1)的影响	将所有被用户或用户应用程序配置的%FC属性值复位。
热重启(%S1=1)的影响	没有影响。
控制器停止的影响	控制器被停止时，%FC继续计数且可以进行参数设置。

超高速计数器功能模块(%VFC)

导言

超高速计数器功能模块(%VFC)可由TwidoSoft配置成执行下面功能之一：

- 加/减计数器
- 加/减双相计数器
- 只加计数器
- 只减计数器
- 频率计

%VFC支持数字输入计数最高频率为20kHz。一体型控制器可以配置一个超高速计数器，模块型控制器最多可配置两个超高速计数器。

专用 **I/O** 分配　　超高速计数器功能模块使用专用输入及辅助输入和输出。这些输入和输出不保留为它们专用。其它功能模块使用这些专用资源时必须考虑它们的分配。下面列表概括了这些分配：

		主要输入		辅助输入		映像输出	
%VFC0	选择使用	输入 IA	输入 IB	IPres	Ica	输出 0	输出 1
	加 / 减 计数器	%I0.0.1	%I0.0.0 (UP=0/DO=1)	%I0.0.2 (1)	%I0.0.3 (1)	%Q0.0.2 (1)	%Q0.0.3 (1)
	加 / 减 双相计数器	%I0.0.1	%I0.0.0 (Pulse)	%I0.0.2 (1)	%I0.0.3 (1)	%Q0.0.2 (1)	%Q0.0.3 (1)
	仅加计数器	%I0.0.1	(2)	%I0.0.2 (1)	%I0.0.3 (1)	%Q0.0.2 (1)	%Q0.0.3 (1)
	仅减计数器	%I0.0.1	(2)	%I0.0.2 (1)	%I0.0.3 (1)	%Q0.0.2 (1)	%Q0.0.3 (1)
	频率计	%I0.0.1	(2)	(2)	(2)	(2)	(2)

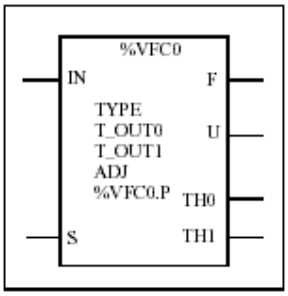
注释：

(1)=可选的　　输入 IA=脉冲输入
(2)=未使用的　　输入 IB=脉冲或 UP/DO
Ipres =预置输入　　UP/DO=加/减
Opt. Use =可选使用

		主要输入		辅助输入		映像输出	
%VFC1	选择使用	输入 IA	输入 IB	IPres	Ica	输出 0	输出 1
	加 / 减 计数器	%I0.0.7	%I0.0.6 (UP=0/DO=1)	%I0.0.5 (1)	%I0.0.4 (1)	%Q0.0.4 (1)	%Q0.0.5 (1)
	加 / 减 双相计数器	%I0.0.7	%I0.0.6 (Pulse)	%I0.0.5 (1)	%I0.0.4 (1)	%Q0.0.4 (1)	%Q0.0.5 (1)
	仅加计数器	%I0.0.7	(2)	%I0.0.5 (1)	%I0.0.4 (1)	%Q0.0.4 (1)	%Q0.0.5 (1)
	仅减计数器	%I0.0.7	(2)	%I0.0.5 (1)	%I0.0.4 (1)	%Q0.0.4 (1)	%Q0.0.5 (1)
	频率计	%I0.0.7	(2)	(2)	(2)	(2)	(2)

	主要输入	辅助输入	映像输出
注释： (1)=可选的 (2)=未使用的 Ipres =预置输入 Opt. Use =可选使用		输入 IA=脉冲输入 输入 IB=脉冲或 UP/DO UP/DO=加/减	

图例　　这里是超高速计数器的一个模块表示：



特性　　下面表格列出了超高速计数器功能模块的特性。

功能	描述	值	**%VFC** 使用	实时访问
当前值 (%VFCi.V)	当前值根据物理输入和选择的功能增加或减少。该值可使用预置输入(%VFCi.S)预置或复位。	0 -> 65535	CM	读
预置值 (%VFCi.P)	仅用于加/减计数功能和只加或只减计数。	0 -> 65535	CM 或 FM	读和写(1)
捕获值 (%VFCi.C)	仅用于加/减计数功能和只加或只减计数。	0 -> 65535	CM	读
计数方向 (%VFCi.U)	被系统设置，该位用于加/减计数功能，表示计数方向： 对于单相加/减计数器，%I0.0.0 决定%VFC0 的方向，%I0.0.6 决定%VFC1 的方向。 对于双相加/减计数器，两个信号的相位的不同决定方向。 对%VFC0，%I0.0为IB专用，%I0.1为IA专用。 对%VFC1，%I0.6为IB专用，%I0.7为IA专用。	0(减计数) 1(加计数)	CM	读
映像输出 0 使能 (%VFCi.R)	使映像输出 0 有效	0(不能) 1(使能)	CM	读和写(2)
映像输出 1 使能 (%VFCi.S)	使映像输出 1 有效	0(不能) 1(使能)	CM	读和写(2)
阈值 S0 (%VFCi.S0)	该字包含阈值 0 的值。其意义在功能模块的配置中定义。注意：该值必须小于%VFCi.S1。	0 -> 65535	CM	读和写(1)
阈值 S1 (%VFCi.S1)	该字包含阈值 0 的值。其意义在功能模块的配置中定义。注意：该值必须大于%VFCi.S0。	0 -> 65535	CM	读和写(1)
频率测量有效 (%VFCi.M)	该位用于决定控制器是否已完成了一个频率测量。	0(无效) 1(有效)	FM	读和写
频率测量时基 (%VFCi.T)	配置项为 100 或 1000 毫秒时基。	1000 或 100	FM	读和写(1)

功能	描述	值	**%VFC** 使用	实时访问
可调节 (Y/N)	配置项，当被选时，允许用户在运行时修改预置值，阈值，和频率测量时基值。	N (不可以) Y (可以)	CM 或 FM	不可以
输入使能 (IN)	用于使当前功能有效或失效。	0 (不可以)	CM 或 FM	读和写(3)
预置输入 (S)	根据配置，状态为 1 时： I 加/减或减计数：用预置值复位当前值。 I 只加计数：将当前值复位到 0。 另外，它也初始化阈值输出的操作，及使用户通过操作显示或用户程序对阈值所作的任何修改生效。	0 或 1	CM 或 FM	读和写
输出溢出 (F)	0到65535 或 从65535到0	0 或 1	CM	读
阈值位 0 (%VFCi.TH 0)	当当前值大于或等于阈值%VFCi.S0时置为1。建议在程序中对该位只测试一次，因为它将被实时更新。用户应用程序对该值使用时的有效性负责。	0 或 1	CM	读
阈值位 1 (%VFCi.TH 1)	当当前值大于或等于阈值%VFCi.S1 时置为 1。建议在程序中对该位只测试一次，因为它将被实时更新。用户应用程序对该值使用时的有效性负责。	0 或 1	CM	读

(1) 只有调节设置为1时才可写。
(2) 只有配置后才可以访问。
(3) 只能通过应用程序，而不是操作显示或活动表编辑器进行读和写访问。
CM = 计数模式
FM = 频率计模式

计数功能描述

超高速计数功能工作最高频率20 kHz，计数范围0到65535。计数脉冲应用于以下几个方面：

表：

功能	描述	%VFC0		%VFC1	
		IA	IB	IA	IB
加/减计数器	脉冲应用于物理输入，当前操作（加计数/减计数）由物理输入 IB 的状态给出。	%I0.0.1	%I0.0.0	%I0.0.7	%I0.0.6
加/减双相计数器	编码器的两相应用于物理输入 IA 和 IB。	%I0.0.1	%I0.0.0	%I0.0.7	%I0.0.6
只加计数器	脉冲应用于物理输入 IA。（IB 未被使用）	%I0.0.1	NA	%I0.0.7	NA
只减计数器	脉冲应用于物理输入 IA。（IB 未被使用）	%I0.0.1	NA	%I0.0.7	NA

与功能模块有关的注意

加计数或减计数操作在脉冲的上升沿完成，且计数模块必须为使能允许。有两个可选输入用于计数模式：Ica和Ipres。Ica用于捕获当前值(%VFCi.V)且将其存储到%VFCi.C。如果配置的话，输入Ica对%VFC0指定为%I0.0.3，对%VFC1指定为%I0.0.4。

当输入Ipres被激活时，当前值在以下几个方面受到影响：

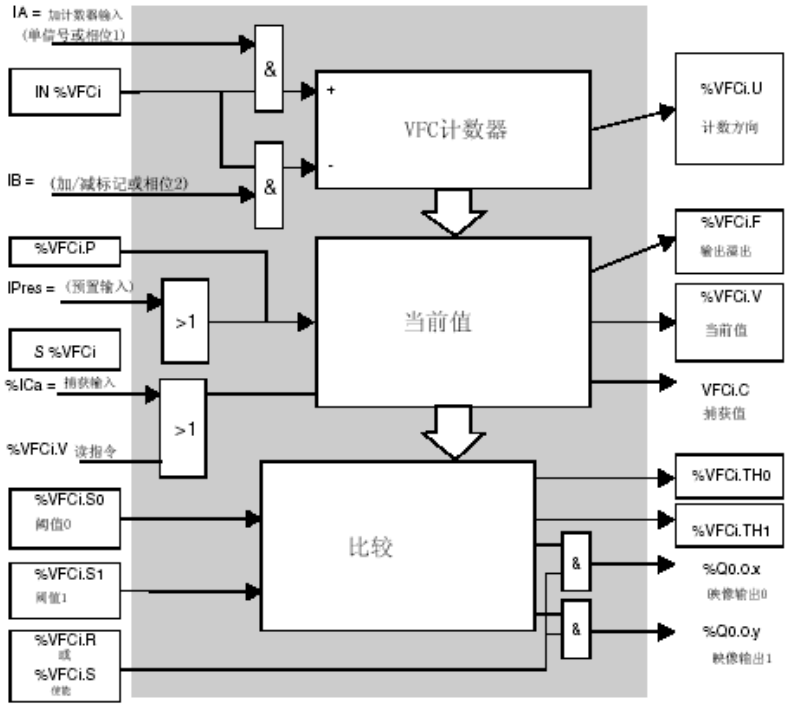
- 对加计数，%VFCi.V被复位到0
- 对减计数，%VFCi.V被%VFCi.P的内容写入
- 对频率计，%VFCi.V和VFCi.M被置为0

警告：%VFCi.F也被置为0。如果配置的话，输入IPres对%VFC0指定为%I0.0.2，对%VFC1指定为%I0.0.5。

与功能模块输出有关的注意

对所有功能，当前值用来与两个阈值(%VFCi.S0和%VFCi.S1)比较。根据比较结果，如果当前值大于或等于对应阈值，则两个位对象(%VFCi.TH0和%VFCi.TH1)被置为1，否则复位到0。映像输出（如果被配置）与这些比较一致被置为1。注意：可以不配置，或配置1个或2个输出。

计数功能图　　下面是一个计数功能图：

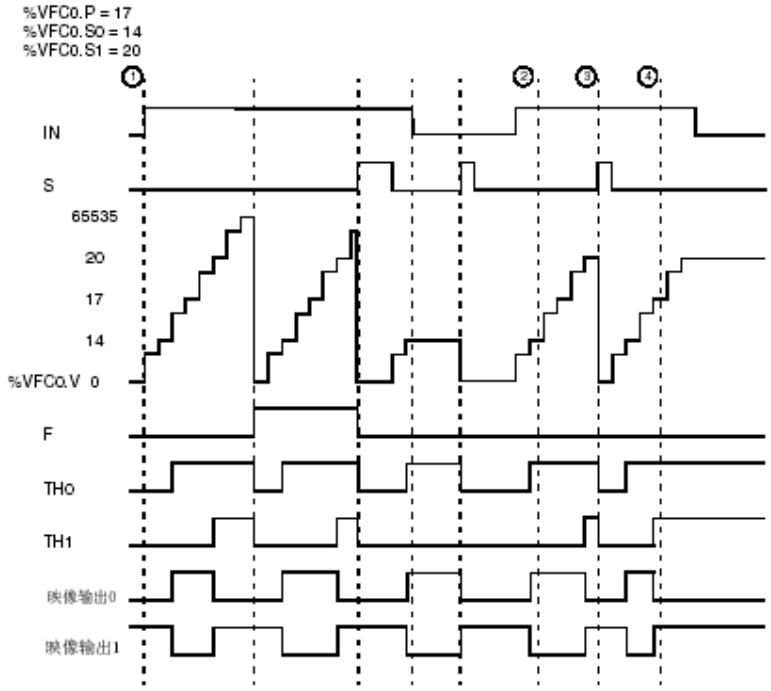


只加计数器操作

下面是一个%VFC只加计数器模式使用示例。此例中下列配置元素已被配置：%VFC0.P预置值是17，%VFC0.S0低阈值是14 ，%VFC0.S1高阈值是20。

映像输出	<%VFC.S0	%VFC0.S0 <=< %VFC0.S1	>= %VFC0.S1
%Q0.0.2		X	
%Q0.0.3	X		X

时序图如下：



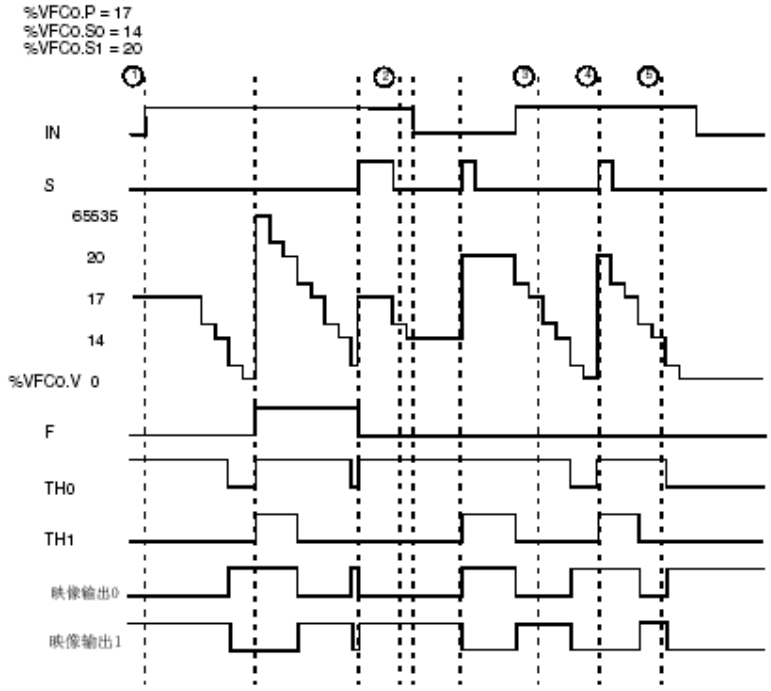
① : %VFC0.U = 1 因为VFC是一个加计数器
② : 改变%VFC0.S1到17
③ : S输入激活使得阈值S1在下次计数时赋为新值
④ : 当前值被捕获到,因此%VFC0.C = 17

只减计数器操作

下面是一个%VFC只减计数器模式使用示例。此例中下列配置元素已被配置：%VFC0.P预置值是17，%VFC0.S0低阈值是14， %VFC0.S1高阈值是20。

映像输出	<%VFC.S0	%VFC0.S0 <=< %VFC0.S1	>= %VFC0.S1
%Q0.0.2	X		X
%Q0.0.3		X	

示例：



① : %VFC0.U = 0 因为VFC是一个减计数器

② ：改变%VFC0.P到20

③ ： 改变%VFC0.S1到17

④ ： S输入激活使得阈值S1在下次计数时赋为新值

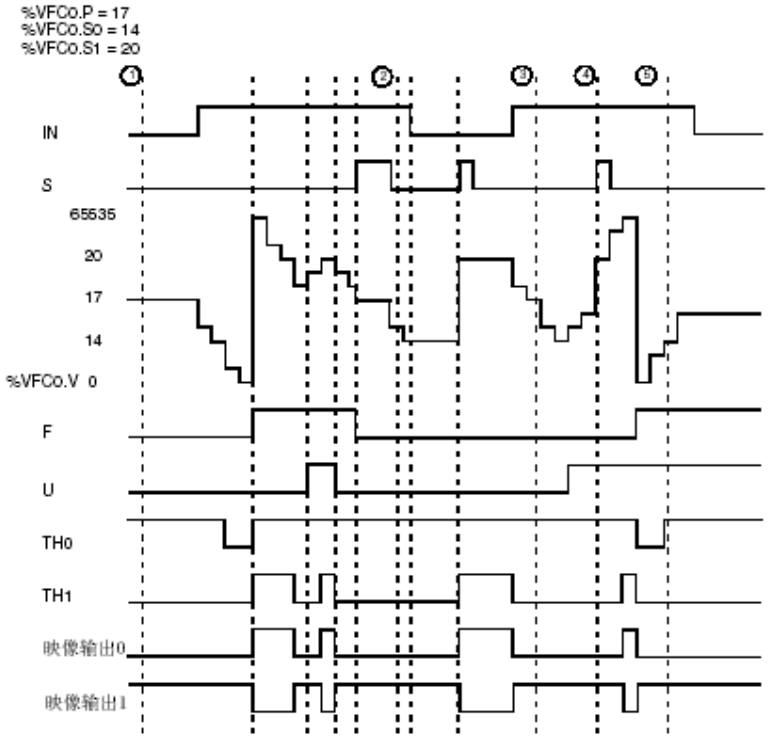
⑤ ： 当前值被捕获到，因此%VFC0.C=17

加-减计数器操作

下面是一个%VFC加-减计数器模式使用示例。此例中下列配置元素已被配置：%VFC0.P预置值是17，%VFC0.S0低阈值是14， %VFC0.S1高阈值是20。

映像输出	VFC.S0	%VFC0.S0 <=< %VFC0.S1	%VFC0.S1
%Q0.0.2			X
%Q0.0.3	X	X	

示例：



① ：输入IN为1且输入S为1

② ：改变%VFC0. P到20

③ ：改变%VFC0. S1到17

④ ：S输入激活使得阈值S1在下次计数时赋为新值

⑤ ：当前值被捕获到，因此%VFC0. C=17

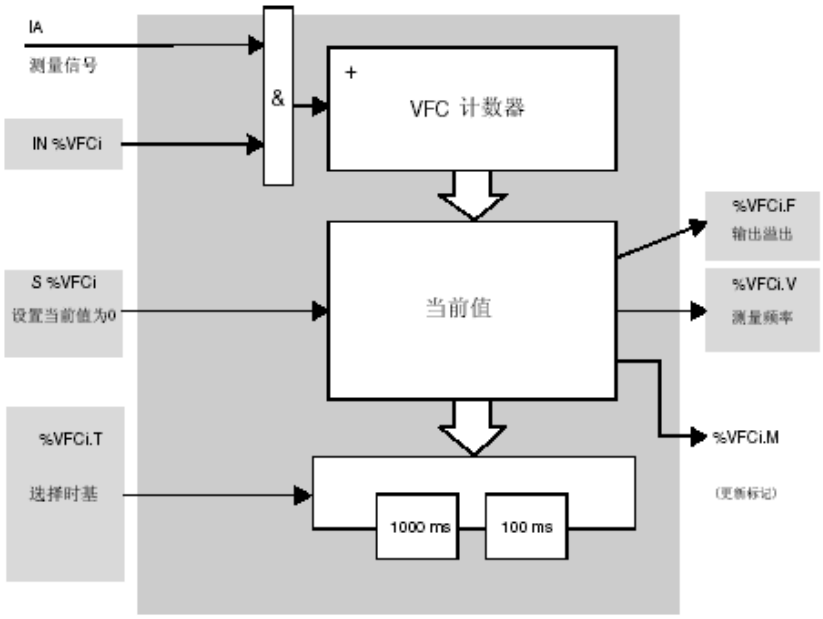
频率计功能描述

%VFC的频率计功能用于测量输入IA上的周期信号的频率，单位为Hz。可测量的频率范围是10 到 20kHz。用户可通过一个新的对象%VFC.T（时基）在2个时基之间选择一个。一个是值100，表示100 ms的时基，一个是值1000，表示1秒的时基。

时基	测量范围	精度	更新
100 ms	100 Hz 到 20 kHz	0.05 % 对于 20 kHz, 10 % 对于 100 Hz	每秒 10 次
1 s	10 Hz 到 20 kHz	0.005 % 对于 20 kHz, 10 % 对于 10 Hz	每秒 1 次

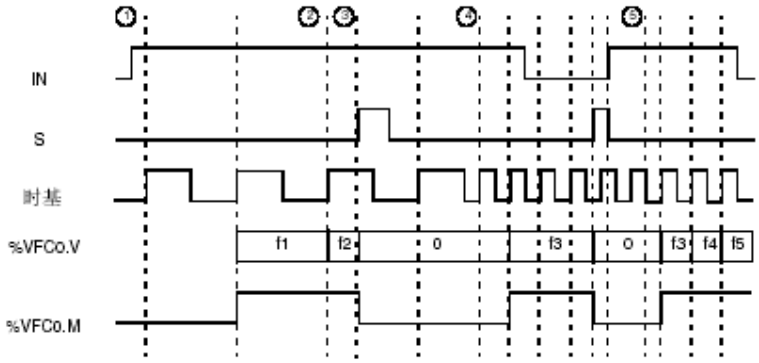
对象%VFC.M（频率测量有效）被置为 1 表示测量完成。

频率计功能图　　下面是频率计功能图：



频率计操作

下面是%VFC 频率计模式使用的时序图示例：



①:第一个频率测量从这里开始。

②:当前频率值被更新。

③:输入 IN 为 1 且输入 S 为 1

④:改变%VFC0.T 到 100ms：这个修改将取消当前测量并从下次测量开始。

⑤: %VFC0.M 由用户置为 0。

特殊情况

下表包含了%VFC 功能模块的特殊操作情况列表：

特殊情况	描述
冷重启(%S0=1)的影响	将所有被用户或用户应用程序配置的%VFC 属性值复位。
热重启(%S1=1)的影响	没有影响。
MCR/MCS 的影响	所有软件输出保持不变。所有配置的映像输出被复位到 0。
控制器停止的影响	%VFC停止其功能且输出保持在当前状态。

发送/接收消息-交换指令（EXCH）

导言

一个 Twido 控制器配置后可与 Modbus 从设备通信，或以字符模式（ASCII）发送和/或接收消息。

TwidoSoft 为这些通信提供了下列功能：

- EXCH 指令用于发送/接收消息
- 交换控制功能模块(%MSG)用于控制数据交换

Twido 控制器在处理 EXCH 指令时使用指定端口的配置协议。每个通信端口可被分配一个不同的协议。通过添加端口号到 EXCH 或%MSG 功能(EXCH1, EXCH2, %MSG1, %MSG2)可以访问通信端口。

EXCH 指令

EXCH 指令允许 Twido 控制器发送和/或接收信息到/从 ASCII 设备。用户定义一个字表(%MWi:L 或 %KWi:L)包含被发送和/或接收的数据（发送和/或接收最多 128 个字节数据）。字表的格式在与每个协议有关的段落被定义。消息交换通过 EXCH 指令完成。

语法

下面是 EXCH 指令的格式：

[EXCHx %MWi:L] 或 [EXCHx %KWi:L]

这里：x = 端口号 (1 or 2)；L = 字在字表中的编号。内部字表%MWi:L 的值为i+L - 255。

Twido控制器必须在第二个交换指令开始之前通过第一个EXCHx指令完成交换。发送几个消息时必须使用%MSG功能模块。

交换控制模块（%MSGx）

导言

注意：%MSGx中的"x"表示控制器端口。

%MSGx 功能模块管理数据交换且具有三个功能：

- 通信错误校验

错误校验核实 EXCH 指令编程的数据长度（字表）足够包含将被发送的消息长度（与字表中的第一个字的低位字节的编程长度比较）。

- 多消息协调

为了确保多条消息发送时的协调性，%MSGx 功能模块提供决定前一条消息何时完成所必需的信息。

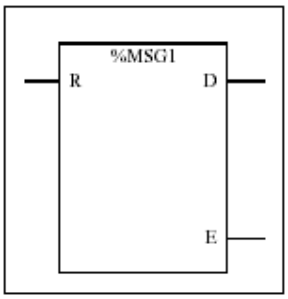
- 优先消息发送

%MSGx功能模块允许当前消息的发送被停止，以保证紧急消息的立即发送。

%MSGx功能模块的编程不是必须的。

图例

下面是一个%MSGx 功能模块示例。



参数

下表列出了%MSGx 功能模块的参数。

参数	标识	值
输入（或指令）复位	R	置为1时，通信重新初始化：%MSGx.E = 0 和%MSGx.D = 1。
通信完成输出	%MSGx.D	状态 1 表示通信在下列情况完成： I 发送结束（如果是发送） I 接收结束（收到结束字符） I 出错 I 模块重启 状态 0 表示请求在处理过程中。
故障（出错）输出	%MSGx.E	状态 1 表示通信在下列情况完成： I 命令错误 I 表配置错误 I 收到不正确的字符（速率，奇偶，等等） I 接收表满（未更新） 状态 0 表示消息长度和连接都正确。

如果使用一个 EXCH 指令时出错，则%MSGx.D 和 %MSGx.E 被置为 1，且系统字%SW63 包含端口 1 的错误代码，%SW64 包含端口 2 的错误代码。见系统字(%SW)，p. 461。

输入复位**(R)**

当输入复位置为 1 时：

- I 处于发送状态的消息被停止。
- I 故障（出错）输出被置为 0。
- I 完成位被置为 1。

现在可以发送一条新的消息。

故障（出错）输出**(%MSGx.E)**

因为通信编程出错，或者是因为消息传送出错时出错输出被置为 1。如果与 EXCH 指令相关的数据模块定义的字节数（字 1 的低位字节）大于 128（十六进制 80），出错输出被置为 1。

如果发送一个Modbus消息到一个Modbus设备中存在问题也将使出错输出被置为1。这种情况下，用户应该检查连线和目的设备对Modbus通信的支持问题。

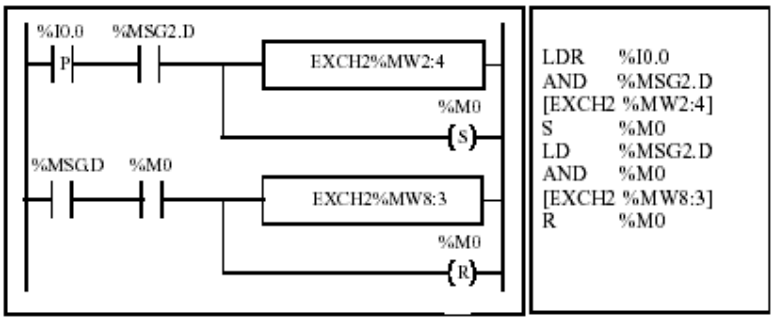
通信完成输出**(%MSGx.D)**

当完成输出被置为 1 时，Twido 控制器准备发送另一个消息。当发送多消息时推荐使用%MSGx.D 位。如果不使用，消息可能被丢失。

多条连续消息的发送

EXCH 指令的执行激活应用程序中的消息模块。消息模块未被激活(%MSGx.D = 1)时消息被发送。如果同一循环中发送多条消息，则只有第一条消息被发送。用户需要使用程序来管理多消息的发送。

在端口 2 上连续发送两条消息的示例：



交换重新初始化

通过激活输入（或指令）R 可以取消一个交换。这个输入初始化通信，将输出%MSGx.E 复位到 0，将输出%MSGx.D 复位到 1。检测到故障时可以对交换重新初始化。

交换重新初始化示例：

特殊情况

下表是%MSGx 功能模块的特殊操作情况。

特殊情况	描述
冷重启(%S0=1)的影响	强制通信重新初始化。
热重启(%S1=1)的影响	没有影响。
控制器停止的影响	如果消息正在发送，则控制器停止其发送并重新初始化输出%MSGx.D和%MSGx.E。

15.2 时钟功能

概览

本节的主题　　本节描述了 Twido 控制器的时间管理功能。

本节包含了哪些内容？　　本节包含了以下主题：

导言

带有实时时钟选件(RTC)的 Twido 控制器具有日历时钟功能，并提供以下功能：

- 调度模块 用于控制预定时间或计划时间执行的动作。
- 时间/日期标记 用于给事件分配时间和日期并测量事件的持续时间。

可通过TwidoSoft软件菜单设置调度模块来访问Twido的日历功能。另外，日历功能可通过程序来设置。如果控制器断电前电池被连续充电6小时以上，则控制器断电后时钟设置可继续工作，最多可达30天。

日历时钟为24小时格式，并且可以考虑闰年情况。

RTC 修正值

RTC 修正值对 RTC 的正确操作是必需的。每个 RTC 单元都写有它自己的修正值。这个值在 TwidoSoft 中可使用控制器操作对话框中的 RTC 配置选项来配置。

调度模块

导言

调度模块用于控制在预定的月，日，时间执行的动作。最多可使用16个调度模块且不需要任何程序输入。

> 注意：检查系统位%S51 确认日历时钟(RTC)选件已安装，见系统位(%S), p. 454。使用调度模块需要 RTC 选件。

参数

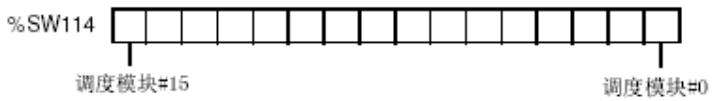
下表列出了调度模块的参数：

参数	格式	功能/范围
调度模块编号	n	n = 0 到 15
配置	确认框	选择这个框配置所选的调度模块编号。
输出位	%Qx.y.z	输出赋值被调度模块%Mi或%Qj.k激活。 当当前日期和时间介于活动周期的开始设置和结束设置之间时，输出被置为1。
开始月	一月到十二月	调度模块的开始月。
结束月	一月到十二月	调度模块的结束月。
开始日期	1 - 31	调度模块的开始日期。
结束日期	1 - 31	调度模块的结束日期。
开始时间	hh:mm	调度模块的开始时间，小时(0 到 23)和分(0 到 23)。
结束时间	hh:mm	调度模块的结束时间，小时(0 到 23)和分(0 到 23)。
星期	星期一到星期日	确认框识别激活的调度模块处于星期几。

调度模块使能

系统字%SW114的位使能（位置为1）或禁止（位置为0）16个调度模块中的每个模块操作。

调度模块在%SW114中的分配：



默认情况下（或冷重启之后）系统字的所有位被置为 1。程序可以选择使用这些位。

调度模块的输出

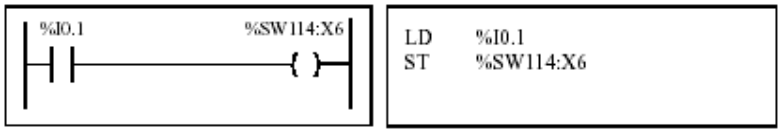
如果同一输出(%Mi 或 %Qj.k)被分配给几个模块，则最后赋值给这个对象的是每个模块的逻辑或结果（同一输入可以有几个“操作范围”）。

示例

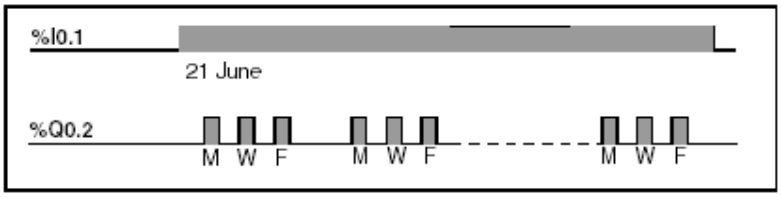
下表显示了用于夏季喷洒程序示例的参数：

参数	值	描述
调度模块	6	调度模块编号 6
输出位	%Qx.y.z	激活输出%Qx.y.z
开始月	六月	六月开始动作
结束月	九月	九月停止动作
开始日期	21	六月 21 日开始动作
结束日期	21	九月 21 日停止动作
星期	星期一，星期三，星期五	星期一，星期三和星期五运行动作
开始时间	21:00	21:00 开始动作
结束时间	22:00	22:00 停止动作

使用下列程序，可以通过一个开关或一个湿度监测器连线到输入%I0.1 禁止该调度模块。



下面时序图是输出%Q0.2 激活显示。



程序中的时间和日期标记

日期和时间都可从系统字%SW50 到%SW53（见系统字(%SW), p. 461）得到。由此，通过当前日期和时间，与包含定值的立即值或字%MWi (或%KWi)进行算术比较，可以实现控制器程序中时间和日期标记。

时间/日期标记

导言

系统字%SW49 到%SW53 包含 BCD 格式的当前日期和时间（见 BCD 码视图，p. 321），这对在外围设备上的显示或发送到外围设备是有用的。这些系统字可用来存储事件的时间和日期（见系统字(%SW), p. 461）。

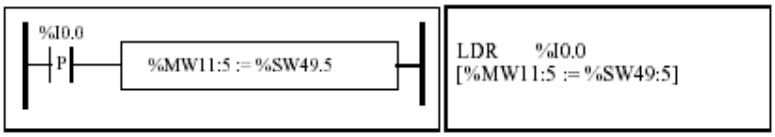
> 注意：日期和时间也可通过可选操作显示模块来设置（见日期时钟时间，p. 203）。

事件日期标记

为了标记事件日期，使用赋值操作，将系统字的内容发送到内部字中，然后处理这些内部字即可（例如，通过 EXCH 指令发送到显示单元）。

编程示例

下面示例显示了怎样用输入%I0.1 的上升沿来标记日期。



一旦检测到一个事件，字表包含：

编码	高位字节	低位字节
%MW11		星期（1）
%MW12	00	秒
%MW13	小时	分
%MW14	月	日
%MW15	世纪	年

> 注意：（1）1 = 星期一, 2 = 星期二, 3 = 星期三, 4 = 星期四, 5 = 星期五, 6= 星期六, 7 = 星期日。

字表示例

2002 年 4 月 19 日，星期一，13:40:30 数据示例：

字	值（十六进制）	意义
%MW11	0001	星期一
%MW12	0030	30 秒
%MW13	1340	13 时，40 分
%MW14	0419	4 月 19 日
%MW15	2002	2002 年

上次停止的日期和时间

系统字%SW54 到%SW57 包含上次停止的日期和时间，且字%SW58 包含显示上次停止原因的代码，均为 BCD 格式（见系统字(%SW), p. 461）。

日期和时间设置

导言

您可以使用下面方法之一更新日期和时间设置：

- TwidoSoft

使用时间设置对话框。此对话框从控制器操作对话框得到。通过从控制器菜单选择控制器操作得到显示。

- 系统字

使用系统字%SW49到%SW53或系统字%SW59。

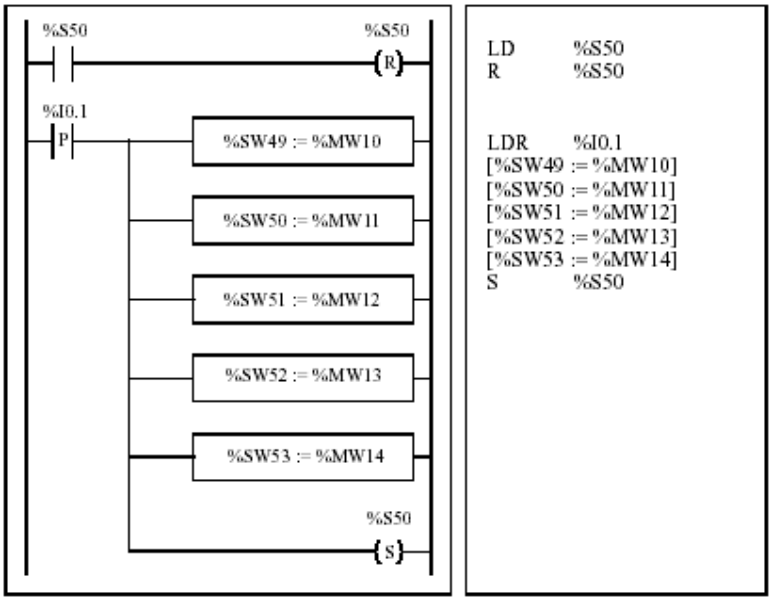
控制器只有在安装RTC选件卡(TWDXCPRTC)的情况下才能更新日期和时间设置。

%SW49 到 %SW53 的使用

为了使用系统字%SW49到%SW53设置日期和时间，位%S50必须被置为1。结果如下：

- 通过内部时钟取消字%SW49到%SW53的更新。
- 发送字%SW49到%SW53的值到内部时钟。

编程示例：



字%MW10 到%MW14 将以 BCD 格式（见 BCD 码视图，p. 321）包含新的日期和时间，且与字%SW49 到%SW53 的编码相对应。

字表必须包含新的日期和时间：

编码	高位字节	低位字节
%MW10		星期（1）
%MW11		秒
%MW12	小时	分
%MW13	月	日
%MW14	世纪	年

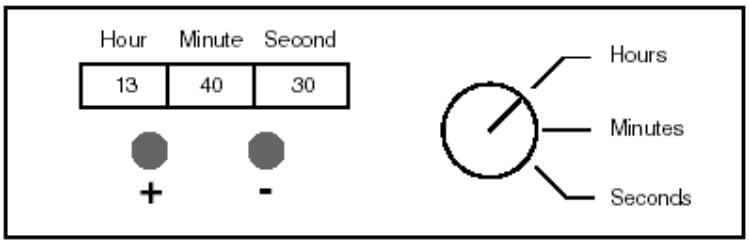
注意:（1）1 = 星期一, 2 = 星期二, 3 = 星期三, 4 = 星期四, 5 = 星期五, 6= 星期六, 7 = 星期日。

使用

更新日期和时间的另一方法是使用系统位%S59 和日期调节系统字%SW59。设置位%S59 为 1 使得可以通过字%SW59（见系统字(%SW), p. 461）调节当前日期和时间。%SW59 在上升沿对每个日期和时间元素增加或减少。

应用示例

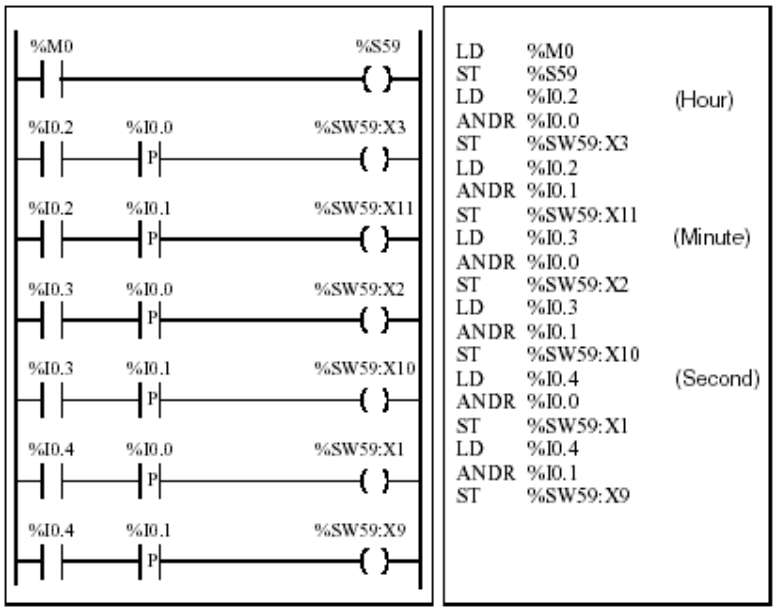
生成下面面板来修改内部时钟的小时，分，和秒。



命令描述：

- 分别使用输入%I0.2，%I0.3，和 %I0.4 使小时/分/秒开关选择需要修改的时间显示。
- 使用输入%I0.0 按按钮"+"增加所选的时间显示。
- 使用输入%I0.1 按按钮"-"减少所选的时间显示。

应用示例　　下面程序从面板读取输入并设置内部时钟。



15.3 PID 功能

概览

本节的主题　　本节描述了 PID 功能的行为，功用和实现。

本节包含了哪些内容？　　本节包含了以下主题：

总的介绍

总述

PID 调节功能是一个 TwidoSoft 编程语言功能。它允许在 Twido1.2 或更高版本的控制器上进行 PID 调节环的编程。

该功能特别适用于：

- 响应连续处理的需要，这些处理需要辅助调节功能（例如：塑料薄膜包装机器，修整处理机器，压力，等等）
- 响应简单调节处理的需要（例如：金属炉，陶炉，小型冷却组，等等）

它的安装非常容易，在屏幕中完成：

- 配置
- 调试

注意：在任何给定的 Twido 自动控制应用中，可以配置的最大 PID 功能数是 14。

重要特性

重要特性如下：

- 模拟输入
- 配置测量的线性转换
- 配置高或低的输入警报
- 模拟或 PWM 输出
- 配置输出的关断
- 配置正向或反向动作

主要调节环

概览

调节环的工作有三个明显的阶段：

- 获得数据：
  - 从处理传感器（模拟，编码器）得到测量值
  - 给定值，一般从控制器的内部变量或 TwidoSoft 活动表中的数据获得
- 执行 PID 调节算法
- 通过离散(PWM)或模拟输出发送相应命令到驱动执行器

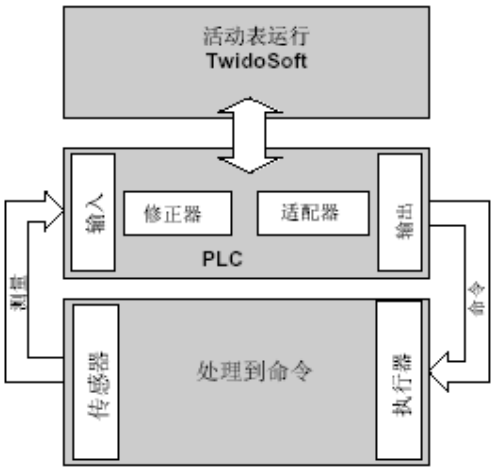
PID算法由以下生成命令信号：

- 输出模块的采样测量值
- 由操作员或程序确定的给定值
- 不同修正参数值

修正信号可被连接到执行器的控制器模拟输出卡直接处理，或者通过控制器的离散输出的PWM调节被处理。

图例

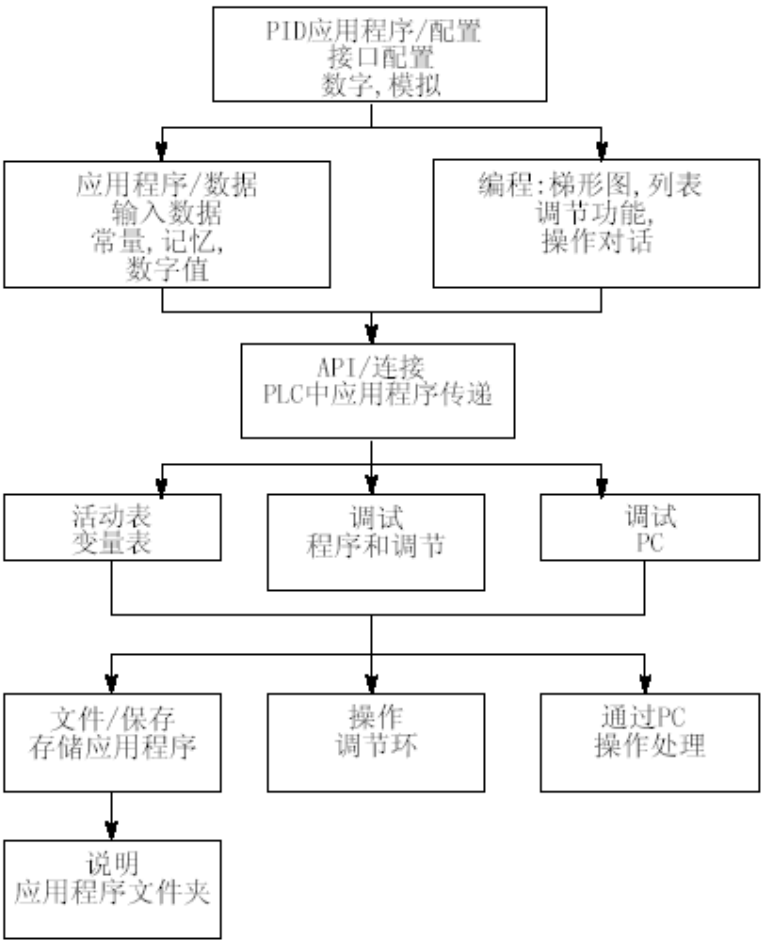
下面是主要调节环的系统图：



主图

下图描述了创建和调试一个调节应用时所执行的全部任务。

注意：顺序定义与您自己的工作方法有关：它仅为一个示例。



兼容和性能

概览

Twido PID 功能适用于 Twido 1.2 和更高版本，这样它的安装对一些硬件和软件具有后面段落所描述的兼容性。
另外，该功能需要性能段落中所提出的资源。

兼容性

Twido PID 功能适用于版本 1.2 或更高版本软件的 Twido。如果您的 Twidos 使用更早版本的软件，您可以更新您的固件以使用 PID 功能。

注意：版本 1.0 的模拟输入/输出模块不需要更新就可以用作 PID 输入/输出模块。

为了在不同的硬件版本上配置和编程，您必须使用版本 **1.2** 的 **TwidoSoft** 软件。

性能

PID 调节环具有如下性能：

描述	时间
环执行时间	0.3 ms

PID 功能的细节特性

概要

PID 功能通过模拟测量和默认[0-10000]格式的给定值完成 PID 修正，并提供相同格式的模拟命令或数字输出的脉宽调制(PWM)。
所有的PID参数在配置它们的窗口中被解释。这里，我们将简要概括可用功能，说明测量值，并描述它们在功能流程图中怎样与PID结合。

注意：为了使用全部范围（最佳分辨率），您可以将连到 PID 测量分支的模拟输入配置成 0-10000 格式。不过，如果您使用默认配置(0-4095)，控制器也将正常运行。

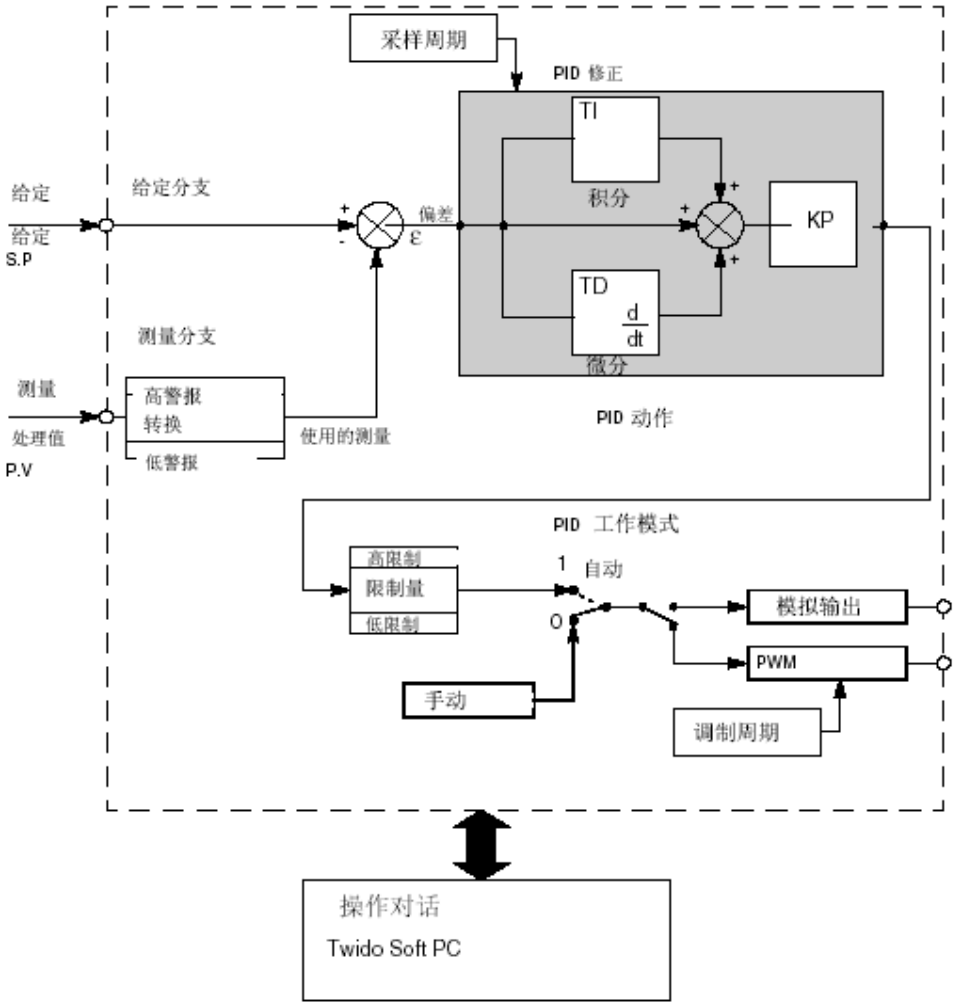
可用功能细节　　下表指出了各种可用功能及其范围：

功能	范围和注释
输入线性转换	允许您将一个 0 到 4095 格式（模拟输入模块分辨率）的值转换为一个-32768 和 32767 之间的值。
比例增益（P）	对系数100，其值介于1和10000之间。它对应一个在0.01和100之间变化的增益值。
积分时间（I）	使用 0.1 秒的时基，其值介于 0 和 32767 之间。它对应一个介于 0 和 3276,7 之间的一个积分时间。
微分时间（D）	使用 0.1 秒的时基，其值介于 0 和 32767 之间。它对应一个介于 0 和 3276,7 之间的一个积分时间。
采样周期	使用 0.01 秒的时基，其值介于 1 和 10000 之间。它对应一个介于 0.01 和 100 之间的一个采样周期。
PWM 输出	使用 0.01 秒的时基，其值介于 1 和 5000 之间。它对应一个介于 0.01 和 50 之间的一个调制周期。
模拟输出	值介于 0 和+10,000 之间
处理变量的高级警报	转换之后设置该警报。如果转换被激活，它被设为一个介于-32768 和 32767 之间的值，否则值介于 0 和 10000 之间。
处理变量的低级警报	转换之后设置该警报。如果转换被激活，它被设为一个介于-32768 和 32767 之间的值，否则值介于 0 和 10000 之间。
输出的高限定值	对一个模拟输出值此限定值介于0和10000之间。当PWM处于活动状态时，此限定值与调制周期的百分比对应。0%对应0，100%对应10000。

功能	范围和注释
输出的低限定值	对一个模拟输出值此限定值介于0和10000之间。当PWM处于活动状态时，此限定值与调制周期的百分比对应。0%对应0，100%对应10000。
手动模式	当手动模式被激活时，输出被赋给一个用户设置的确定值。这个输出值介于0和10000之间（PWM输出为0 到100%）。
正向或反向动作	正向或反向都可以，且直接对输出动作。

注意：对于上表描述的每个功能怎样工作的更为深入的解释，请参见下面图。

工作原理　　下图呈现了 PID 功能的工作原理。



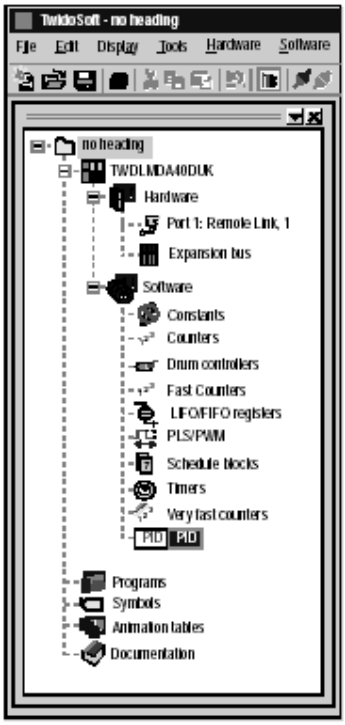
注意：使用的参数的描述见上页表格和配置屏幕。

怎样访问 PID 配置

概览　　下面图描述了怎样访问 TWIDO 控制器的 PID 配置屏幕。

过程

下表描述了访问 PID 配置屏幕的过程：

步骤	动作
1	确认处于离线模式。
2	打开浏览器。 结果： 
3	双击 **PID**。 结果：PID 配置窗口被打开并默认显示 **General**（见 PID 功能总表，p. 408）表。 注意：您也可以右击**PID**并选择**Edit**选项或从菜单选择 **Software PID**，或使用**Program Configuration Editor PID Icon**菜单，如果使用后一种方法，选择PID并点击 **Magnifying glass**图标选择一种PID。

概览

当您从浏览器打开 PID，您打开的是 PID 配置窗口。这个窗口允许您：

- 配置每个 TWIDO PID，
- 调试每个 TWIDO PID，

当您打开这个屏幕时，根据情况：

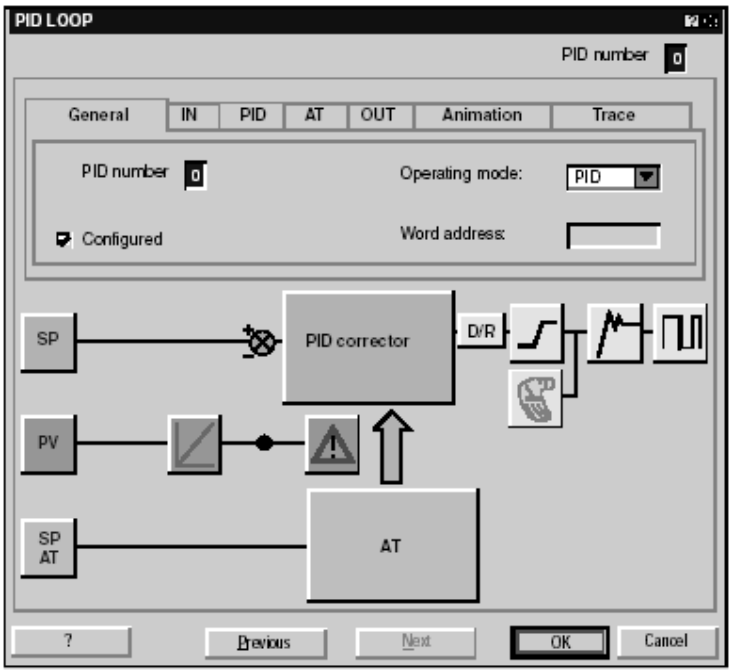
- 如果是离线模式：您将进入默认的 **General** 表且可以进行参数配置，
- 如果是在线模式：您将进入 **Animation** 表且可以进行参数调试和调节。

注意：不能访问变成灰色的表和地方。变灰是因为它们只在以后的版本中被提供（例如自动调节），或因为该当前的活动模式（离线或在线）不允许您访问这些参数。

下面段落描述了 **General** 表。

PID 功能的 **General** 表

下面屏幕用于输入总的 PID 参数。



描述

下表是对您可以定义的设置的描述。

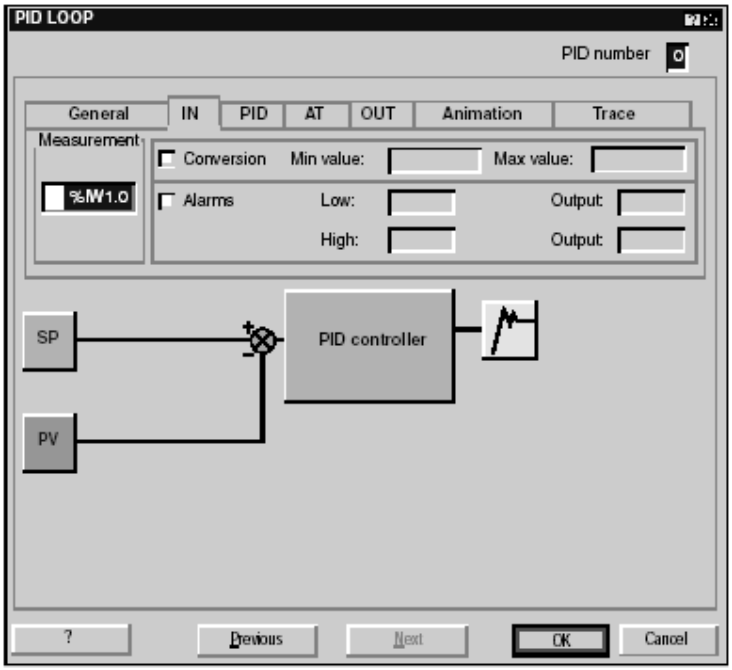
区域	描述
PID number	在这里指定您想配置的 PID 编号。 值介于 0 到 13 之间，每个应用最多 14 个 PID。
Configured	必须选中此框才能配置 PID。否则屏幕不能执行动作，且 PID 即便存在与应用程序中也不能使用。
Operating mode	在这里指定期望的操作模式。 在 Twido 的这个版本，可选选项只有 PID。以后的版本将有其它可选选项。
Word address	这个版本没有与自动调节相关的选项。
图	这个图允许您观看您配置的各种可能的PID。

PID 功能 IN 表

概览　　该表用于输入 PID 输入参数。

注意：它在离线模式下被访问。

PID 功能 **IN** 表　　下面屏幕用于输入 PID 输入参数。



描述

下表是对您可以定义的设置的描述。

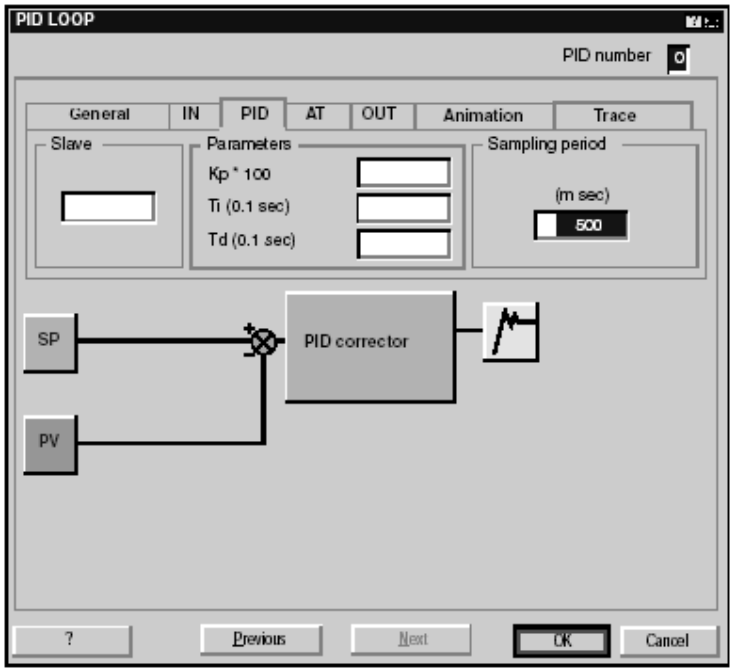
区域	描述
PID number	在这里指定您想配置的 PID 编号。 值介于 0 到 13 之间，每个应用最多 14 个 PID。
Measurement	在这里指定变量，变量包含被控制的处理值。 默认范围是 0 到 10000。您可以输入一个内部字 (%MW0 到 %MW1499) 或一个模拟输入 (%IWx.0 到 %IWx.1)。
Conversion	如果您想转换指定为 PID 输入的处理值则选中此框。 如果选中此框，则可以访问**Min value** 和 **Max value**。 转换是线性的且将一个0 和 10,000之间的值转换为一个最小值和最大值介于-32768 和+32767之间的值。
Min value **Max value**	在这里指定 PID 可能的最小和最大输入值。然后 PID 自动转换处理值确保不超过这些最小和最大值。 注意：**Min value** 必须小于 **Max value**。 **Min value**或**Max value**可以是内部字(%MW0 到 %MW1499)，内部常量(%KW0 到 %KW1499)或介于-32768 和 +32767之间的值。
Alarms	如果您想激活高或低输入值的警报则选中此框。 注意：警报值应相对于转换阶段之后获得的处理值。当转换被激活时它们必须介于 **Min value** 和 **Max value** 之间。否则它们必须介于 0 和 10000 之间。
Low Output	在 **LOW** 域指定低警报值。 该值可以是一个内部字(%MW0 到%MW1499)，一个内部常量(%KW0 到%KW1499)或一个直接值。**Output**必须包含到达低限定值时被置为1的位的地址。**Output**可以是一个内部位(%M0到%M255)或一个输出(%Qx.0到%Qx.32)。
High Output	在 **High** 域指定高警报值。 该值可以是一个内部字(%MW0 到%MW1499)，一个内部常量(%KW0 到%KW1499)或一个直接值。**Output** 必须包含到达高限定值时被置为 1 的位的地址。**Output** 可以是一个内部位(%M0 到%M255)或一个输出(%Qx.0 到%Qx.32)。
图	这个图允许您观看您配置的各种可能的PID。

PID 功能 PID 表

概览　　该表用于输入内部 PID 参数。

注意：它在离线模式下被访问。

PID 功能 PID 表　下面屏幕用于输入内部 PID 参数。



描述

下表是对您可以定义的设置的描述。

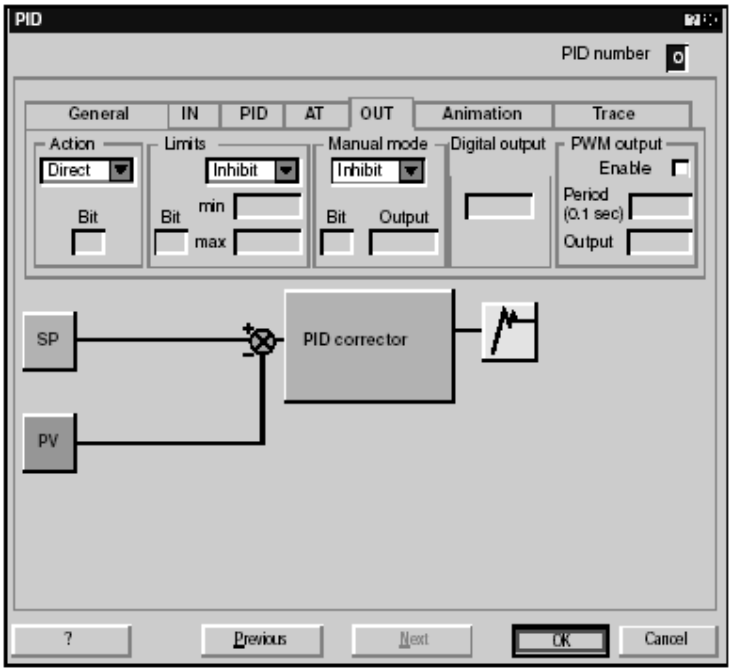
区域	描述
PID number	在这里指定您想配置的 PID 编号。 值介于 0 到 13 之间，每个应用最多 14 个 PID。
Slave	在这里指定 PID 设定值。该值可以是一个内部字(%MW0 到 %MW1499)，一个内部常量(%KW0 到 %KW1499)或一个直接值。 当转换被禁止时该值介于 0 和 10000 之间。否则它必须介于转换的 **Min value** 和 **Max value** 之间。
Kp * 0.01	在这里指定乘上0.01后的比例系数。 该值可以是一个内部字(%MW0 到 %MW1499)，一个内部常量(%KW0 到 %KW1499)或一个直接值。 它必须介于0和10000 之间。
TI (0.1 sec)	在这里指定时基为0.1秒的积分动作系数。 该值可以是一个内部字(%MW0 到 %MW1499)，一个内部常量(%KW0 到 %KW1499)或一个直接值。 它必须介于0和32,767之间。
Td (0.1 sec)	在这里指定时基为0.1秒的微分动作系数。 该值可以是一个内部字(%MW0 到 %MW1499)，一个内部常量(%KW0 到 %KW1499)或一个直接值。 它必须介于0和32,767之间。
Sampling period	在这里指定时基为0.01秒的PID采样周期。 该值可以是一个内部字(%MW0 到 %MW1499)，一个内部常量(%KW0 到 %KW1499)或一个直接值。 它必须介于0和127之间。
图	这个图允许您观看您配置的各种可能的PID。

PID 功能 OUT 表

概览　　该表用于输入 PID 输出参数。

注意：它在离线模式下被访问。

PID 功能 **OUT** 表　　下面屏幕用于输入 PID 输出参数。



描述

下表是对您可以定义的设置的描述。

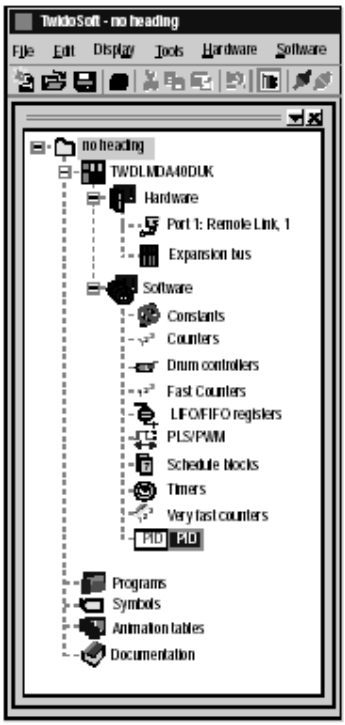
区域	描述
PID number	在这里指定您想配置的 PID 编号。 值介于 0 到 13 之间，每个应用最多 14 个 PID。
Action	在这里指定处理的 PID 动作类型。有三个选项可选：**Direct**, **Reverse** 或 **bit address**。 如果您选择了 **bit address**，您可以通过程序使用相关位修改这个类型，这个相关位是一个内部位(%M0 到%M255)或一个输入(%Ix.0 到%Ix.32)。 当该位被置为 1 时动作是正向的，否则是反向的。
Limits Bit	在这里指定您是否为PID输出设置限定值。有三个选项可选：**Enable**, **Disable** 或 **bit address**。 如果您选择了**bit address**，您可以通过程序修改相关位使得可以(位为1)或不可以(位为0)限制管理，这个相关位是一个内部位(%M0到%M255)或一个输入(%Ix.0到%Ix.32)。
Min **Max**	在这里指定 PID 输出的阈值。 注意：**Min** 必须总是小于 **Max**。 **Min** 或 **Max** 可以是内部字(%MW0 到 %MW1499)，内部常量(%KW0 到 %KW1499)或介于 1 和 10,000 之间的值。
Manual mode Bit **Output**	在这里指定您是否切换PID到手动模式。有三个选项可选：**Enable**, **Disable** 或 **bit address**。 如果您选择了 **bit address**，您可以通过程序修改相关位使得切换到手动模式(位为 1)或切换到自动模式(位为 0)，这个相关位是一个内部位(%M0 到%M255)或一个输入(%Ix.0 到%Ix.32)。 当PID处于手动模式时，手动模式的**Output**必须包含您想赋给数字输出的值。 这个**Output**可以是一个字(%MW0到%MW1499)或一个直接值。
Digital output	在这里指定PID输出为自动模式。 **Digital output**可以是一个字(%MW0到%MW1499)或一个模拟输出(%QWx.0)。
PWM output enabled **Period (0.1s)** **Output**	如果您想使用 PID 的 PWM 功能，请选中此框。 在**Period (0.1s)**指定调制周期。这个周期必须介于1和500之间，且可以是一个内部字(%MW0 到 %MW1499)或内部常量(%KW0 到 %KW1499)。 指定PWM输出位作为**Output**值。这可以是一个内部位(%M0到%M255)或一个输出(%Qx.0到%Qx.32)。
图	这个图允许您观看您配置的各种可能的PID。

怎样访问 PID 调试

概览　　下面段落描述了怎样访问 TWIDO 控制器的 PID 调试屏幕。

过程

下表描述了访问 PID 调试屏幕的过程：

步骤	动作
1	确认处于在线模式。
2	打开浏览器。 结果： 
3	双击 **PID**。 结果：PID 配置窗口被打开并默认显示 **Animation**（见 PID 功能活动表，p. 418）表。 注意：您也可以右击**PID**并选择**Edit**选项或从菜单选择 **Software PID**，或使用**Program Configuration Editor PID Icon**菜单，如果使用后一种方法，选择PID并点击 **Magnifying glass**图标选择一种PID。

PID 功能活动表

概览

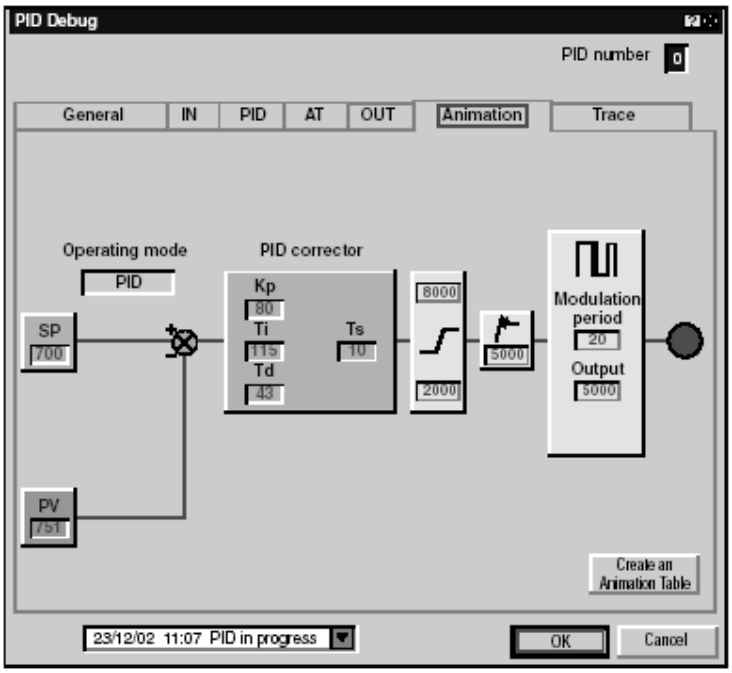
该表用于调试 PID。
该图表取决于您已经创建的 PID 控制类型。只有已配置元素可被显示。

该显示是动态的。活动连接显示为红色，非活动连接显示为黑色。

注意：它在在线模式下被访问。

PID 功能活动表

下面屏幕用于 PID 显示和调试。



描述

下表描述了窗口的各个区域。

区域	描述
PID number	在这里指定您想调试的 PID 编号。 值介于 0 到 13 之间，每个应用最多 14 个 PID。
Operating mode	这个区域显示当前 PID 工作模式。
Create an Animation Table	点击，创建一个文件，包含图表显示的所有变量，使得您在线修改它们和调试PID。
列表	该列表位于屏幕的底部，允许您实时查看PID最近的15个状态。该列表在每个状态变化时被更新，并指示变化的日期和时间以及当前状态。

PID 功能跟踪表

概览

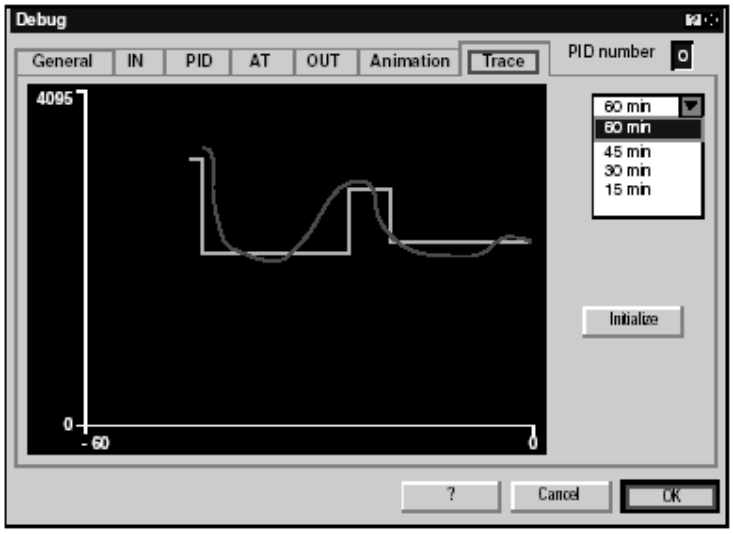
该表允许您查看 PID 操作，并对其行为方式进行调节。

一旦显示调试窗口，则该跟踪图就开始工作。

注意：它在在线模式下被访问。

PID 功能跟踪表

下面屏幕用于 PID 控制查看。



描述

下表描述了窗口的各个区域。

区域	描述
PID number	在这里指定您想查看的 PID 编号。 值介于 0 到 13 之间，每个应用最多 14 个 PID。
Chart	这个区域显示了 **setpoint** 和 **process value** 图。 水平坐标标度(X)使用窗口右上角菜单决定。 垂直坐标标度使用 PID 输入配置值（带或不带转换）决定。它自动被优化以获得图形的最佳显示。
Horizontal axis scale menu	这个菜单允许您修改水平轴的标度。您可以从4个值中选择：15, 30, 45 或 60分。
Initialize	这个按钮清除图表并重新开始跟踪图。

15.4 浮点指令

概览

本节的主题

本节描述了 TwidoSoft 语言中的高级浮点（见浮点和双字对象，p.31）指令。

比较和赋值指令描述见数字运算，p. 303。

本节包含了哪些内容？

本节包含了以下主题：

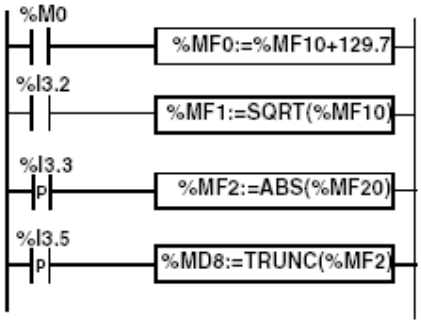
浮点算术指令

总述　　这些指令用于执行两个操作数或一个操作数的算术运算。

+	两个操作数相加	**SQRT**	一个操作数的平方根
-	两个操作数相减	**ABS**	一个操作数的绝对值
*****	两个操作数相乘	**TRUNC**	浮点值的整数部分
/	两个操作数相除	**EXP**	自然指数
LOG	以 10 为底的对数	**EXPT**	实数的实求幂
LN	自然对数		

结构

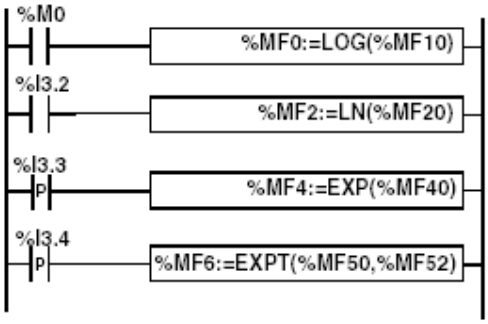
梯形图语言



指令列表语言

```
LD %M0
[%MF0:=%MF10+129.7]

LD %I3.2
[%MF1:=SQRT(%MF10)]

LDR %I3.3
[%MF2:=ABS(%MF20)]

LDR %I3.5
[%MD8:=TRUNC(%MF2)]
```

梯形图语言



指令列表语言

```
LD %M0
[%MF0:=LOG(%MF10]

LD %I3.2
[%MF2:=LN(%MF20)]

LDR %I3.3
[%MF4:=EXP(%MF40)]

LDR %I3.4
[%MF6:=EXPT(%MF50,%MF52)]
```

语法

浮点算术指令运算符和语法

运算符	语法
+, - *, /	Op1:=Op2 运算符 Op3
SQRT, ABS, TRUNC, LOG, EXP, LN	Op1:=运算符(Op2)
EXPT	Op1:=运算符 (Op2,Op3)

注意：当您执行两个浮点数的加法或减法运算时，这两个操作数必须满足条件：？？？，这里 Op1>Op2。如果这个条件不能被满足，结果将等于操作数 1(Op1)。这种现象只是在单独运算的情况下出现的极少结果，结果错误非常小（2^{-24}），但是它在重复计算时有着不可预料的结果。

例如指令**%MF2:= %MF2 + %MF0**被不确定重复这种情况。如果初始条件是%MF.0 = 1.0和%MW2= 0，值%MF2将被锁在16777216。

因此我们提醒您在对重复计算编程时要特别小心。如果您相对这类计算进行编程，则客户应用程序要对截断误差进行管理。

浮点算术指令的操作数：

运算符	操作数 1**(Op1)**	操作数 2**(Op2)**	操作数 3**(Op3)**
+, - *, /	%MFi	%MFi, %KFi, 立即值	%MFi, %KFi, 立即值
SQRT, ABS, LOG, EXP, LN	%MFi	%MFi, %KFi	[-]
TRUNC	%MDi	%MFi, %KFi	[-]
EXPT	%MFi	%MFi, %KFi	%MFi, %KFi

使用规则

- 浮点运算和整数值运算不能直接混合。转换操作（见整数转换指令<->浮点，p. 433）将这些格式互相转换。
- 系统位%S18 的管理方法与整数运算（见整数算术指令，p. 313）相同，字%SW17（见系统字(%SW), p. 461）指示错误原因。
- 当函数的操作数是无效数（例如：对一个负数求对数）时，将产生一个不定或无穷大的结果，并将位%S18 变为 1，字%SW17 指示错误原因。

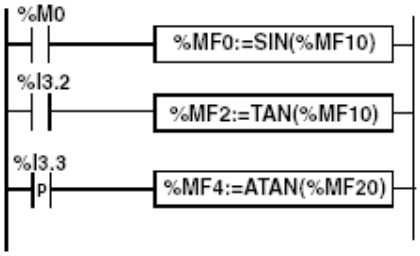
三角函数指令

总述　　这些指令使得用户可以执行三角函数运算。

SIN	一个角度（弧度）的正弦，	**ASIN**	反正弦（结果在 $-\frac{\pi}{2}$ 和 $\frac{\pi}{2}$ 之间）
COS	一个角度（弧度）的余弦，	**ACOS**	反余弦（结果在 0 和 π 之间）
TAN	一个角度（弧度）的正切，	**ATAN**	反正切（结果在 $-\frac{\pi}{2}$ 和 $\frac{\pi}{2}$ 之间）

结构

梯形图语言



指令列表语言

```
LD %M0
[%MF0:=SIN(%MF10)]

LD %I3.2
[%MF2:=TAN(%MF10)]

LDR %I3.3
[%MF4:=ATAN(%MF20)]
```

结构文本语言

```
IF %M0 THEN
 %MF0:=SIN(%MF10);
END_IF;
IF %I3.2 THEN
 %MF2:=TAN(%MF10);
END_IF;
IF %I3.3 THEN
 %MF4:=ATAN(%MF20);
END_IF;
```

语法

三角函数运算指令的运算符，操作数和语法

运算符	语法	操作数 1 **(Op1)**	操作数 2 **(Op2)**
SIN, COS, TAN, ASIN, ACOS, ATAN	Op1:= 运 算 符 (Op2)	%MFi	%MFi, %KFi

使用规则

- 当函数的操作数是无效数（例如：对一个大于 1 的数求反余弦）时，将产生一个不定或无穷大的结果，并将位%S18 变为 1，字%SW17（见系统字(%SW), p. 461）指示错误原因。
- 函数 SIN/COS/TAN 允许介于-4096π和4096π之间的角度参数，但是它们的精度将随着离开区间-2π和$+2\pi$逐渐下降，这是因为参数在运算之前要以2π为模进行操作，这将带来不精确性。

转换指令

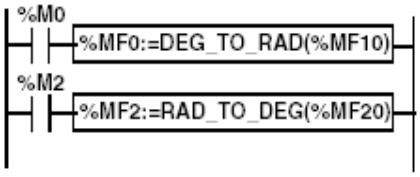
总述

这些指令用于执行转换操作。

DEG_TO_RAD	将角度转换为弧度，结果值介于 0 和？？之间
RAD_TO_DEG	将弧度转换为角度，结果值介于 0 和 360 度之间

结构

梯形图语言



指令列表语言

```
LD %M0
[%MF0:=DEG_TO_RAD(%MF10)]

LD %M2
[%MF2:=RAD_TO_DEG(%MF20)]
```

结构文本语言

```
IF %M0 THEN
 %MF0:=DEG_TO_RAD(%MF10);
END_IF;
IF %M2 THEN
 %MF2:=RAD_TO_DEG(%MF20);
END_IF;
```

语法

转换指令的运算符，操作数和语法

运算符	语法	操作数 1 **(Op1)**	操作数 2 **(Op2)**
DEG_TO_RAD **RAD_TO_DEG**	Op1:= 运 算 符 (Op2)	%MFi	%MFi, %KFi

使用规则

被转换的角必须介于-737280.0 和+737280.0 之间（对 DEG_TO_RAD 转换）或介于-4096π和4096π之间（对 RAD_TO_DEG 转换）。

对于这些范围之外的值，结果将显示为+ 1.#NAN，位%S18 和%SW17:X0 将被置为 1。

整数转换指令<->浮点

总述

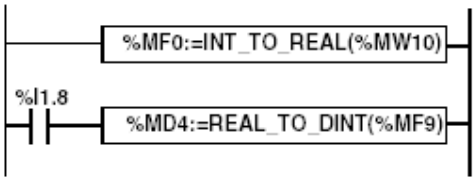
提供了四个转换指令。

整数转换指令列表<->浮点：

INT_TO_REAL	转换一个整数字-->浮点
DINT_TO_REAL	转换双整数字-->浮点
REAL_TO_INT	浮点转换-->整数字（结果为最接近的代数值）
REAL_TO_DINT	浮点转换-->双整数字（结果为最接近的代数值）

结构

梯形图语言



指令列表语言

```
LD TRUE
[%MF0:=INT_TO_REAL(%MW10)]

LD I1.8
[%MD4:=REAL_TO_DINT(%MF9)]
```

结构文本语言

```
%MF0:=INT_TO_REAL(%MW10);
IF %I1.8 THEN
 %MD4:=REAL_TO_DINT(%MF9);
END_IF;
```

语法

INT_TO_REAL	转换一个整数字-->浮点
DINT_TO_REAL	转换双整数字-->浮点
REAL_TO_INT	浮点转换-->整数字（结果为最接近的代数值）
REAL_TO_DINT	浮点转换-->双整数字（结果为最接近的代数值）

运算符和语法（转换一个整数字-->浮点）：

运算符	语法
INT_TO_REAL	Op1=INT_TO_REAL(Op2)

操作数（转换一个整数字-->浮点）：

操作数 1 (Op1)	**操作数 2 (Op2)**
%MFi	%MWi,%KWi

示例：整数字-->浮点：147 --> 1.47e+02

运算符和语法（转换双整数字-->浮点）：

运算符	语法
DINT_TO_REAL	Op1=DINT_TO_REAL(Op2)

操作数（转换双整数字-->浮点）：

操作数 1 (Op1)	**操作数 2 (Op2)**
%MFi	%MDi,%KDi

示例：双整数字转换-->浮点：68905000 --> 6.8905e+07

运算符和语法（浮点转换-->整数字或双整数字）：

运算符	语法
REAL_TO_INT	Op1=Operator(Op2)
REAL_TO_DINT	

操作数（浮点转换-->整数字或双整数字）：

类型	**操作数 1 (Op1)**	**操作数 2 (Op2)**
字	%MWi	%MFi, %KFi
双字	%MDi	%MFi, %KFi

示例：

浮点转换-->整数字：5978.6 --> 5979

浮点转换-->双整数字：-1235978.6 --> -1235979

注意：如果实数转换到整数（或实数转换到双整数字）时浮点值超出字（或双字）的限制，位%S18 将被置为 1。

取整精度

IEEE 754 标准为浮点运算定义了 4 种取整模式。
上面指令采用的是“最接近值取整”模式：“如果可取的最接近值与理论值距离相等，则取其低有效位为 0 的那个值。”
某些情况下，取整的结果可以是省略值或超额值。

例如：
取整 10.4 -> 10
取整 11.5 -> 12

15.5 对象表指令

概览

本节的主题

本节描述了有关下面表的特殊指令：

- 双字，
- 浮点对象。

表的赋值指令已在章节“基本指令”中描述（见字，双字和浮点表赋值，p. 309）。

本节包含了哪些内容？

本节包含了以下主题：

表求和功能

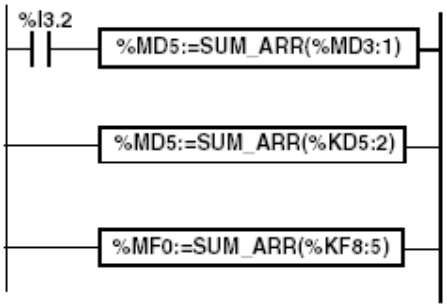
总述

SUM_ARR 功能将一个对象表的所有元素加在一起：

- 如果表由双字组成，则结果以双字格式给出
- 如果表由浮点字组成，则结果以浮点字格式给出

结构

梯形图语言



指令列表语言

```
LD %I3.2
[%MD5:=SUM_ARR(%MD3:1)]
%MD5:=SUM_ARR(%KD5:2)
%MF0:=SUM_ARR(%KF8:5)
```

语法

表求和指令语法：

Res:=SUM_ARR(Tab)

表求和指令参数

类型	结果**(res)**	表**(Tab)**
双字表	%MDi	%MDi:L,%KDi:L
浮点字表	%MFi	%MFi:L,%KFi:L

注意：当结果不在表操作数对应的有效双字格式范围内时，系统位 %S18 将被置为 1。

示例

```
%MD5:=SUM(%MD30:4)
```

当 %MD30=10, %MD31=20, %MD32=30, %MD33=40 时，

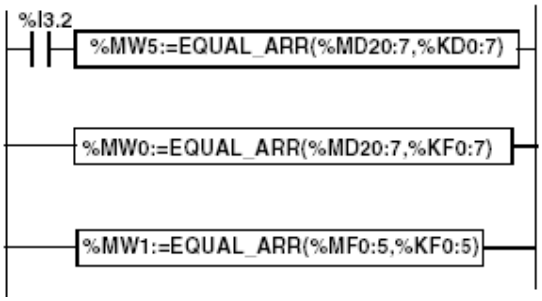
%MD5=10+20+30+40=100

总述

EQUAL _ARR 功能对两个表的元素逐个进行比较。
如果找到不同，则第一个不同元素的序列号以一个字的形式返回，否则返回值等于-1。
对整个表执行比较操作。

结构

梯形图语言



指令列表语言

```
LD %I3.2
[%MW5:=EQUAL_ARR(%MD20:7,KD0:7)]
```

结构文本语言

```
%MW0:=EQUAL_ARR(%MD20:7,%KF0:7)

%MW1:=EQUAL_ARR(%MF0:5,%KF0:5)
```

语法

表比较指令语法：

```
Res:=EQUAL_ARR(Tab1,Tab2)
```

表比较指令参数

类型	结果**(res)**	表**(Tab1 和 Tab2)**
双字表	%MWi	%MDi:L,%KDi:L
浮点字表	%MWi	%MFi:L,%KFi:L

注意：

- 表的长度和类型必须相同。

示例

```
%MW5:=EQUAL_ARR(%MD30:4,%KD0:4)
```

2个表的比较：

Rank	Word Table	Constant word tables	Difference
0	%MD30=10	%KD0=10	=
1	%MD31=20	%KD1=20	=
2	%MD32=30	%KD2=60	Different
3	%MD33=40	%KD3=40	=

字%MW5的值是2（第一个不同元素的序列号）

表查找指令

总述

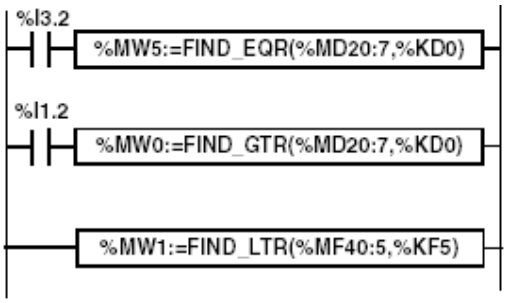
有 3 个查找函数：

- **FIND_EQR**：在双字或浮点字表中查找与给定值相等的第一个元素的位置
- **FIND_GTR**：在双字或浮点字表中查找比给定值大的第一个元素的位置
- **FIND_LTR**：在双字或浮点字表中查找比给定值小的第一个元素的位置

如果找到符合条件的第一个元素，则指令的结果等于它的序列号，否则等于-1。

结构

梯形图语言



指令列表语言

```
LD %I3.2
[%MW5:=FIND_EQR(%MD20:7,KD0)]
LD %I1.2
[%MW0:=FIND_GTR(%MD20:7,%KD0)]
%MW1:=FIND_LTR(%MF40:5,%KF5)
```

语法

表查找指令语法：

函数	语法
FIND_EQR	Res:=函数(Tab,Val)
FIND_EQR	
FIND_LTR	

浮点字和双字表查找指令参数

类型	结果**(res)**	表**(Tab)**	值**(val)**
浮点字表	%MWi	%MFi:L,%KFi:L	%MFi,%KFi
双字表	%MWi	%MDi:L,%KDi:L	%MDi,%KDi

示例

```
%MW5:=FIND_EQR(%MD30:4,%KD0)
```

查找表中=%KD0=30的第一个双字的位置：

Rank	Word Table	Result
0	%MD30=10	-
1	%MD31=20	-
2	%MD32=30	Value (val), rank
3	%MD33=40	-

表最大值和最小值查找功能

总述

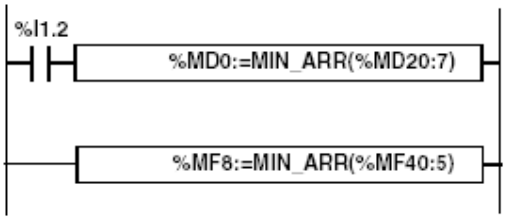
有 2 个查找函数：

- **MAX_ARR**：查找双字或浮点字表中的最大值
- **MIN_ARR**：查找双字或浮点字表中的最小值

这些指令的结果等于表中发现的最大值（或最小值）。

结构

梯形图语言



指令列表语言

```
LD %I1.2
[%MD0:=MIN_ARR(%MD20:7)]
%MF8:=MIN_ARR(%MF40:5)
```

语法

表最大值和最小值查找指令语法：

函数	语法
MAX_ARR	Res:=函数(Tab)
MIN_ARR	

表最大值和最小值查找指令参数：

类型	结果**(Res)**	表**(Tab)**
双字表	%MDi	%MDi:L,%KDi:L
浮点字表	%MFi	%MFi:L,%KFi:L

表中某个值的出现次数

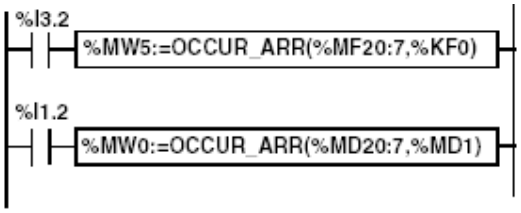
总述

这个查找函数：

- **OCCUR_ARR**：查找双字或浮点字表中与给定值相等的元素的个数。

结构

梯形图语言



指令列表语言

```
LD %I3.2
[%MW5:=OCCUR_ARR(%MF20:7,%KF0)]
LD %I1.2
[%MW0:=OCCUR_ARR(%MD20:7,%MD1)
```

语法

函数语法：

函数	语法
OCCUR_ARR	Res:=函数(Tab,Val)

指令参数

类型	结果**(res)**	表**(Tab)**	值**(val)**
双字表	%MWi	%MDi:L,%KDi:L	%MDi,%KDi
浮点字表	%MWi	%MFi:L,%KFi:L	%MFi,%KFi

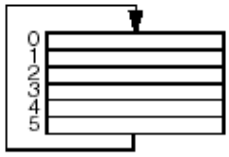
表循环移动功能

总述

有2个移动函数：

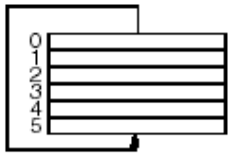
- **ROL_ARR**：在浮点字表中从上到下循环移动n个位置

ROL_ARR函数图例



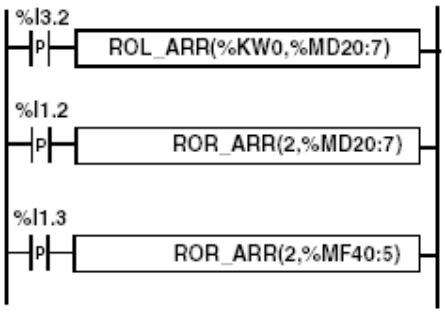
- **ROR_ARR**：在浮点字表中从下到上循环移动n个位置

ROR_ARR函数图例



结构

梯形图语言



指令列表语言

```
LDR %I3.2
[ROL_ARR(%KW0,%MD20:7)]
LDR %I1.2
  [ROR_ARR(2,%MD20:7)]
LDR %I1.3
[ROR_ARR(2,%MF40:5)]
```

语法

浮点字或双字表**ROL_ARR** 和 **ROR_ARR**循环移动指令语法：

函数	语法
ROL_ARR	函数(n,Tab)
ROR_ARR	

浮点字表 **ROL_ARR** 和 **ROR_ARR** 循环移动指令参数：

类型	位置数**(n)**	表**(Tab)**
浮点字表	%MWi，立即值	%MFi:L
双字表	%MWi，立即值	%MDi:L

注意：如果 n 值为负或零，则不执行移动。

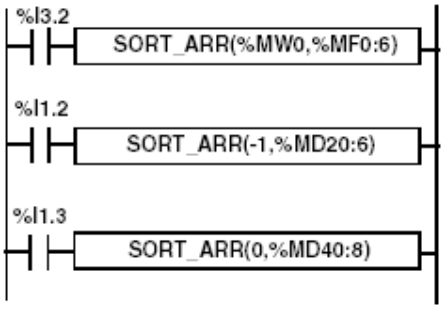
总述

有下面的排序函数可用：

- **SORT_ARR**：对一个双字或浮点字表的元素按照升序或降序进行排序并将结果存储在相同的表中。

结构

梯形图语言



指令列表语言

```
LD %I3.2
[SORT_ARR(%MW20,%MF0:6)]
LD %I1.2
[SORT_ARR(-1,%MD20:6)]
LD %I1.3
[SORT_ARR(0,%MF40:8)
```

语法

表排序函数语法：

函数	语法
SORT_ARR	函数(direction,Tab)

- "direction"参数给出排序顺序：direction > 0 表示升序；direction < 0 表示降序，direction = 0 表示不进行排序。
- 结果（已排序表）返回到 Tab 参数（要排序的表）。

表排序函数参数：

类型	排序顺序	表**(Tab)**
双字表	%MWi，立即值	%MDi:L
浮点字表	%MWi，立即值	%MFi:L

表长度计算功能（不被支持）

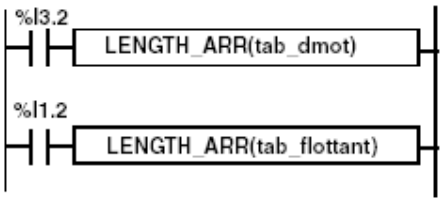
总述

表长度计算函数如下：

- **LENGTH_ARR**: 通过元素个数计算双字或浮点字表的长度

结构

梯形图语言



指令列表语言

```
LD %I3.2
[LENGTH_ARR(tab_dmot)]
LD %I1.2
[LENGTH_ARD(tab_flottant)
```

语法

表长度计算函数语法：

函数	语法
LENGTH_ARR	结果=函数(Tab)

表长度计算函数参数

表**(Tab)**	结果**(Res)**
双字: %MDi, %KDi	%MWi
浮点字: %MFi, %KFi	%MWi

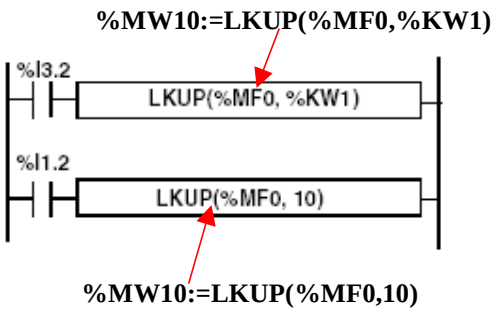
注意：表参数将纯粹为符号对象。

总述

LKUP函数用于通过各浮点之间的线性查补对表中浮点值进行图形化表示。此函数在表中的浮点宽度可变。
用户必须指定表中需要以内插值替换的浮点数数量。

结构

梯形图语言



指令列表语言

LD %I3.2
[%MW10:=LKUP(%MF0,%KW1)]
LD %I1.2
[%MW10:=LKUP(%MF0,10)]

删除

插值

表中浮点插补算法如下：

$$Y = Y_i + ((Y_{i+1} - Y_i)/(X_{i+1} - X_i))*(X - X_i)$$
for $X_i \le X \le X_{i+1}$ where i = 1 ... (m-1)

语法

函数语法：

函数	语法
LKUP	结果=LKUP(Tab, Nb)

函数操作数：

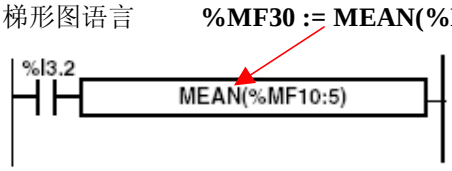
表**(Tab)**	结果**(Res)**	**Nb**
%MFi	%MWi	立即值, %MWi, %KWi

操作数"Nb"表示表中需要考虑的浮点个数。它必须是一个偶数。

浮点表求均值功能

总述　　**MEAN** 函数用于计算一个浮点表中给定数目的值的均值。

结构　　梯形图语言　　**%MF30 := MEAN(%MF10:5)**



指令列表语言

LD　%I3.2

[%MF30:=MEAN(%MF10:5)]

删除

语法　　浮点表均值计算函数语法：

函数	语法
MEAN	结果=函数(Op1)

浮点表给定 L 个数的值的计算函数的参数：

操作数**(Op1)**	结果**(Res)**
%MFi:L, %KFi:L	%MFi

16

概览

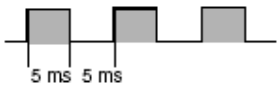
本章的主题　　本章提供了用于创建 Twido 控制器控制程序的系统位和系统字的概述。

本章包含了哪些内容？　　本章包含了以下主题：

系统位(%S)

导言　　下面部分提供了有关系统位功能及怎样控制它们的详细信息。

详细描述　　下面表格提供了有关系统位及怎样控制它们的概述。

系统位	功能	描述	初始状态	控制
%S0	冷启动	一般置为 0，它被下列情况置为 1： ❙ 电源恢复且数据丢失（电池故障）， ❙ 用户程序或动态数据表表， ❙ 操作显示模块。 该位在第一次完全扫描时被置为 1。它在下一次扫描前被系统置为 0。	0	S 或 U->S
%S1	热启动	一般置为 0，它被下列情况置为 1： ❙ 电源恢复且数据备份， ❙ 用户程序或动态数据表表， ❙ 操作显示模块。 该位在完全扫描结束时被系统置为 0。	0	S 或 U->S
%S4 %S5 %S6 %S7	时基: 10 ms 时基: 100 ms 时基: 1 s 时基: 1 min	状态变化率由内部时钟测量。它们与控制器扫描不同步。 示例：%S4  	-	S
%S8	连线测试	初始置为 1，该位用于控制器“非配置”状态测试连线： ❙ 置为 1，输出复位， ❙ 置为 0，连线测试被授权。	1	U
%S9	复位输出	一般置为 0。它可以被程序或终端（通过动态数据表编辑器）置为 1： ❙ 状态为 1 时，若控制器处于运行模式则输出被强制到 0， ❙ 状态为 0 时，输出被正常更新。	0	U
%S10	I/O 故障	一般置为 1。当检测 I/O 故障时该位被系统置为 0。	1	S
%S11	看门狗溢出	一般置为 0。当程序执行时间（扫描时间）超过最大扫描时间（软件看门狗）时该位被系统置为 1。 看门狗溢出导致控制器进入暂停状态。	0	S

系统位	功能	描述	初始状态	控制
%S12	PLC 处于运行模式	该位表示控制器处于运行状态。系统在控制器运行时将该位置为 1。在停止，初始，或其它状态时置为 0。	0	S
%S13	运行的第一个循环	一般置为 0，在控制器变为运行状态后的第一个扫描过程中被系统置为 1。	1	S
%S17	容量超出	一般置为 0，它在下列情况被系统置为 1： • 无符号算术运算（取余）时装载溢出， • 在循环或移动操作时。系统表示输出位为 1。它必须由用户程序在每次可能产生溢出的操作之后测试，溢出发生后由用户复位到 0。	0	S->U
%S18	算术溢出或错误	一般置为 0。它在进行 16 位的运算时出现溢出的情况下被置为 1： • 单字长度下结果大于+ 32 767 或小于- 32 768， • 双字长度下结果大于+ 2 147 483 647 或小于- 2 147 483 648， • 浮点结果大于 3.402824E+38 或小于- 3.402824E+38， • 被 0 除， • 对负数求平方根， • BTI 或 ITB 转换无意义：BCD 值超出限制。 它必须由用户程序在每次可能产生溢出的操作之后测试，溢出发生后由用户复位到0。	0	S->U
%S19	扫描周期溢出（周期扫描）	一般置为0，该位在扫描周期溢出（扫描时间大于用户在配置中定义或在%SW0中编程的周期）的情况下被系统置为1。该位由用户复位到0。	0	S->U
%S20	索引溢出	一般置为 0，它在索引对象的地址小于 0 或大于对象的最大范围时被置为 1。 它必须由 它必须由用户程序在每次可能产生溢出的操作之后测试，然后在溢出发生后复位到 0。	0	S->U

系统位	功能	描述	初始状态	控制
%S21	GRAFCET 初始化	一般置为 0，它被下列情况置为 1： ❙ 冷重启，%S0=1， ❙ 用户程序，且只能在预处理程序部分，使用 Set 指令(S %S21)或设置线圈-(S)- %S21， ❙ 终端。 状态为 1 时，它导致 GRAFCET 初始化。活动步被停止且激活初始步。它在 GRAFCET 初始化之后被系统复位到 0。	0	U->S
%S22	GRAFCET 复位	一般置为 0，它只能由程序的预处理置为 1。 状态为 1 时它导致全部 GRAFCET 的活动步停止。它在顺序处理开始执行时由系统复位到 0。	0	U->S
%S23	预置和冻结 GRAFCET	一般置为 0，它只能在预处理程序模块由程序置为 1。 置为 1 时，它使 GRAFCET 的预置生效。维持该位在 1 将冻结 GRAFCET（冻结图表）。它在顺序处理开始执行时由系统复位到 0 以保证 GRAFCET 表从冻结状态变为移动状态。	0	U->S
%S24	操作显示	一般置为0，该位可被用户置为1。 ❙ 状态为0时，操作显示模块正常工作， ❙ 状态为1时，操作显示模块被冻结，保持当前显示不变，不能闪烁，且停止输入键处理。	0	U->S
%S31	事件标志	一般置为0，该位可被用户置为1。 ❙ 状态为0时，事件可被执行， ❙ 状态为1时，事件不能被执行且排队等待。 该位可由用户和系统（在冷重启情况下）置为0。	0	U->S

系统位	功能	描述	初始状态	控制
%S38	允许事件进入事件队列	一般置为 0，该位可被用户置为 1。 ▌ 置为 0 时，一旦检测到事件就将它们放置到事件队列， ▌ 置为 1 时，事件不能进入事件队列。 该位可由用户和系统（在冷重启情况下）置为 0。	0	U->S
%S39	事件队列饱和	一般置为 0，该位可被用户置为 1。 ▌ 置为 0 时，所有事件都被报告， ▌ 置为 1 时，至少一个事件被丢失。 该位可由用户和系统（在冷重启情况下）置为 0。	0	U->S
%S50	使用字 %SW50 到 53 更新日期和时间	一般置为 0，该位可被程序或操作显示置为 1。 ▌ 置为 0 时，日期和时间均可读， ▌ 置为 1 时，日期和时间均不可被更新。	0	U->S
%S51	日历时钟状态	一般置为 0，该位可被程序或操作显示置为 1。 ▌ 置为 0 时，日期和时间是一致的， ▌ 置为 1 时，日期和时间必须由用户初始化。 当该位置为 1 时，日期时钟的时间数据无效。日期和时间可能从未配置过，电池可能电压不够，或控制器修正常量可能无效。 状态1到状态0的转变强制写入修正常量到RTC。	0	U->S
%S59	使用字 %S59 更新日期和时间	一般置为 0，该位可被程序或操作显示置为 1。 ▌ 置为 0 时，不能管理系统字 %SW59， ▌ 置为 1 时，日期和时间根据%SW59 设置的控制位的上升沿增加或减少。	0	U
%S69	用户 STAT LED 显示	置为0时，STAT LED关断。 置为1时，STAT LED打开。	0	U
%S70	AS-I 总线更新数据	该位在每个控制器循环或在AS-I总线扫描循环结束时由系统置为1。 上电时，它表示所有数据至少已被更新一次，因而它很重要。 该位由用户复位到0。	0	S->U

系统位	功能	描述	初始状态	控制
%S73	改变为 AS-i 总线保护模式	一般置为 0，该位由用户置为 1 以切换到 AS-i 总线保护模式。这之前该位必须已经被置为 1。 该位只用于连线系统测试。它在控制器中没有应用。	0	S
%S74	保存 AS-i 总线配置	一般置为 0，该位由用户置为 1 以保存 AS-i 总线当前配置。 该位只用于连线系统测试。它在控制器中没有应用。	0	S
%S95	恢复存储字	当需要保存内部字到内部 EEPROM 时可以设置该位。完成后系统将该位置回 0 且存储的存储字数置于%SW97。	0	U
%S96	备份程序完成	该位可在任何时刻被读取（被程序读或调节时读），特别是在冷启动或热重启之后。 I 置为 0 时，备份程序无效。 I 置为 1 时，备份程序有效。	0	S
%S97	保存%MW 完成	该位可在任何时刻被读取（被程序读或调节时读），特别是在冷启动或热重启之后。 I 置为 0 时，保存%MW 无效。 I 置为 1 时，保存%MW 有效。	0	S
%S100	TwidoSoft 通信电缆连接	显示TwidoSoft通信电缆是否已连接。 I 置为1时，没有连接TwidoSoft通信电缆或连接的是TwidoSoft。 I 置为0时，TwidoSoft远程连接电缆已连接。	-	S
%S110	远程连接交换	该位被程序或终端复位到0。 I 对主机，置为1表示所有的远程连接交换（仅远程I/O）完成。 I 对从机，置为1表示和主机的交换完成。	0	S->U
%S111	单一远程连接交换	I 对主机，置为0表示单一远程连接交换完成。 I 对主机，置为1表示单一远程连接交换处于活动状态。	0	S
%S112	连接远程连接	I 对主机，置为0表示远程连接处于非活动状态。 I 对主机，置为1表示远程连接处于活动状态。	0	U

系统位	功能	描述	初始状态	控制
%S113	远程连接配置/操作	▌ 对主机或从机，置为 0 表示远程连接配置/操作完成。 ▌ 对主机，置为 1 表示其远程连接配置/操作出错。 ▌ 对从机，置为 1 表示其远程连接配置/操作出错。	0	S->U
%S118	远程 I/O 出错	一般置为 1。当远程连接检测到 I/O 故障时该位被置为 0。	1	S
%S119	本地 I/O 出错	一般置为 1。当远程连接检测到 I/O 故障时该位被置为 0。%SW118 决定故障种类。当故障消除时复位到 1。	1	S

表缩写描述　　缩写表：

缩写	描述
S	被系统控制
U	被用户控制
U->S	由用户置为 1，由系统复位到 0
S->U	由系统置为 1，由用户复位到 0

系统字(%SW)

导言　　下面部分提供了有关系统字功能及怎样控制它们的详细信息。

详细描述　　下面表格提供了有关系统字功能及怎样控制它们的详细信息。

系统字	功能	描述	控制
%SW0	控制器扫描周期（周期任务）	通过用户程序在动态数据表编辑器中修改配置中定义的控制器扫描周期。	U
%SW6	控制器状态	控制器状态： 0 = 没有配置 2 = 停止 3 = 运行 4 = 中断	S

系统字	功能	描述	控制
%SW7	控制器状态	▪ 位[0]：备份/恢复处理： ▪ 置为 1 表示备份/恢复在处理中， ▪ 置为 0 表示备份/恢复完成或不能进行。 ▪ 位[1]：控制器的配置完成： ▪ 置为 1 表示配置完成。 ▪ 位[3..2] EEPROM 状态位： ▪ 00 = 没有卡 ▪ 01 = 32 Kb EEPROM 卡 ▪ 10 = 64 Kb EEPROM 卡 ▪ 11 = 保留给将来使用 ▪ 位[4]：RAM 中应用程序与 EEPROM 不同： ▪ 置为 1 表示 RAM 应用程序与 EEPROM 不同。 ▪ 位[5]：RAM 应用程序与卡中的不同： ▪ 置为 1 表示 RAM 应用程序与卡中的不同。 ▪ 位[6]不被使用（状态为 0） ▪ 位[7]：控制器保留： ▪ 置为 1 表示保留。 ▪ 位[8]：应用程序处于写模式： ▪ 置为 1 表示应用程序受保护。 ▪ 位[9]不被使用（状态为 0） ▪ 位[10]：第二个串口已安装： ▪ 置为 1 表示已安装。 ▪ 位[11]：第二个串口类型：(0 = EIA RS-232, 1 = EIA RS-485)： ▪ 置为 0 = EIA RS-232 ▪ 置为 1 = EIA RS-485 ▪ 位[11]：内部存储中应用程序有效： ▪ 置为 1 表示应用程序有效。 ▪ 位[13]：卡中应用程序有效： ▪ 置为 1 表示应用程序有效。 ▪ 位[14]：RAM 中应用程序有效： ▪ 置为 1 表示应用程序有效。 ▪ 位[15]：准备执行： ▪ 置为 1 表示已准备执行。	S
%SW11	软件看门狗值	包含看门狗的最大值。值(10到500ms)由配置定义。	U

系统字	功能	描述	控制
%SW17	浮点运算默认状态	当浮点算术运算检测到出错时，位%S18 被置为 1 且%SW17 的缺省状态根据下面代码得到更新： • 位 [0]：无效运算，结果不是一个数 (1.#NAN or -1.#NAN)。 • 位 1：非标准操作数（介于-1.175494e-38 和 1.175494e-38 之间），结果截断为 0.0。 • 位 2：被 0 除，结果为无穷大（-1.#INF 或 1.#INF） • 位 3：结果绝对值大于+3.402824e+38，结果为无穷大（-1.#INF 或 1.#INF）	S 和 U
%SW18-%SW19	100 ms 绝对定时计数器	计数器工作使用这两个字： • **%SW18** 表示低有效字， • **%SW19** 表示高有效字。	S 和 U
%SW30	上一次扫描时间	显示上一次控制器扫描循环的执行时间（ms）。 注意：这个时间对应一个扫描循环从开始（输入请求）到结束（输出更新）的时间。	S
%SW31	最大扫描时间	显示自上一次冷启动以来最长的控制器扫描循环的执行时间（分钟）。 注意：这个时间对应一个扫描循环从开始（输入请求）到结束（输出更新）的时间。	S
%SW32	最小扫描时间	显示自上一次冷启动以来最短的控制器扫描循环的执行时间（分钟）。 注意：这个时间对应一个扫描循环从开始（输入请求）到结束（输出更新）的时间。	S
%SW48	事件数	显示自上一次冷启动以来执行了多少个事件。 注意：设置为 0（在应用程序装载和冷启动之后），每个事件的执行导致其增加。	S

系统字	功能	描述		控制
%SW49 %SW50 %SW51 %SW52 %SW53	实时时钟(RTC)	RTC 功能：字包含当前日期和时间值（BCD 格式）：		S 和 U
		%SW49	xN 星期（N=1 表示星期一）	
		%SW50	00SS 秒	
		%SW51	HHMM 时和分	
		%SW52	MMDD 月和日	
		%SW53	CCYY 世纪和年	
		当位**%S50** 为 0 时这些字由系统控制。 当位**%S50** 置为 1 时这些字可由用户程序或终端写入。		
%SW54 %SW55 %SW56 %SW57	上一次停止的日期和时间	系统字包含上一次电源故障或控制停止时的日期和时间（BCD 格式）：		S
		%SW54	SS 秒	
		%SW55	HHMM 时和分	
		%SW56	MMDD 月和日	
		%SW57	CCYY 世纪和年	
%SW58	上一次停止的代码	显示上一次停止的原因代码：		S
		1＝	运行/停止输入沿	
		2＝	因软件故障停止（控制器扫描溢出）	
		3＝	停止命令	
		4＝	电源故障	
		5＝	因硬件故障停止	

<table>
<tr><th>系统字</th><th>功能</th><th>描述</th><th>控制</th></tr>
<tr><td>%SW59</td><td>调节当前日期</td><td>调节当前日期。
包含两个调节当前日期的 8 位设置。
该操作在位的上升沿执行。位%S59 使该字可用。
<table>
<tr><th>增加</th><th>减少</th><th>参数</th></tr>
<tr><td>位 0</td><td>位 8</td><td>星期</td></tr>
<tr><td>位 1</td><td>位 9</td><td>秒</td></tr>
<tr><td>位 2</td><td>位 10</td><td>分</td></tr>
<tr><td>位 3</td><td>位 11</td><td>时</td></tr>
<tr><td>位 4</td><td>位 12</td><td>天</td></tr>
<tr><td>位 5</td><td>位 13</td><td>月</td></tr>
<tr><td>位 6</td><td>位 14</td><td>年</td></tr>
<tr><td>位 7</td><td>位 15</td><td>世纪</td></tr>
</table></td><td>U</td></tr>
<tr><td>%SW60</td><td>RTC 修正</td><td>RTC 修正值</td><td>U</td></tr>
<tr><td>%SW63</td><td>EXCH1 模块错误代码</td><td>如果使用 EXCH 模块时出错，输出位%MSG.D 和 %MSG.E 将变为 1。该系统字包含最近一次出错时收到的错误代码。如果有两个错误，您必须先解决最近报告的错误。
I 0：没有错误，交换正确
I 1：发送表太大
I 2：发送表太小
I 3：表太小
I 4：接收表溢出
I 5：定时超出
I 6：发送出错
I 7：ASCII 命令错误（仅 ASCII 模式）
I 8：所选端口不可配置/不可用
I 9：接收出错（仅 ASCII 模式）
I 10：表%Kwi 被禁止
每当使用 EXCH 模块时该字被置为 0。</td><td>S</td></tr>
<tr><td>%SW64</td><td>EXCH2 模块错误代码</td><td>与%SW63 相同</td><td>S</td></tr>
</table>

系统字	功能	描述	控制
%SW67	控制器功能和类型	包含下列信息： I 控制器类型位[0 -11] I 8B0 = TWDLCAA10DRF I 8B1 = TWDLCAA16DRF I 8B2 = TWDLMDA20DUK/DTK I 8B3 = TWDLCAA24DRF I 8B4 = TWDLMDA40DUK/DTK I 8B6 = TWDLMDA20DRT I 位 12,13,14,15 不被使用 = 0	S

系统字	功能	描述	控制
%SW73 和 %SW74	AS-I 系统状态	位[0]：置为 1 表示配置完成。	S 和 U
		位[1]：置为 1 表示数据交换被激活。	
		位[2]：置为 1 表示模块处于离线模式。	
		位[3]：置为 1 表示 ASI_CMD 指令结束。	
		位[4]：置为 1 表示 ASI_CMD 指令正在处理。	
%SW76 到 %SW79	减计数器 1-4	这 4 个字用作 1 ms 定时器。如果它们包含一个正值则它们每毫秒被系统减计数。这使得 4 个减计数器以毫秒为单位减计数，等于工作范围为 1 ms 到 32767 ms。置位 15 为 1 可以停止减计数。	S 和 U

系统字	功能	描述	控制
%SW96	应用程序和%MW 的保存/恢复功能的命令和/或诊断。	▪ 位[0]：表示%MW 存储字必须保存到EEPROM： ▪ 置为 1 表示需要备份， ▪ 置为 0 表示备份处理没有完成。 ▪ 位[1]：该位由固件设置表示保存完成： ▪ 置为 1 表示备份完成， ▪ 置为 0 表示一个新的备份请求被要求。 ▪ 位[2]：备份错误，参考位 8, 9, 10 和 14 以得到更多信息： ▪ 置为 1 表示出现错误， ▪ 置为 0 表示一个新的备份请求被要求。 ▪ 位[6]：置为 1 表示控制器包含一个空的应用程序。 ▪ 位[8]：表示%SW97 中指定的%MWs 数比应用程序中配置的%MWs 数大： ▪ 置为 1 表示检测到错误， ▪ 位[9]：表示%SW97 中指定的%MWs 数比 TwidoSoft 中任何应用程序中可以配置的%MWs 最大数大。 ▪ 置为 1 表示检测到错误， ▪ 位[10]：内部 RAM 和内部 EEPROM 不同（1=是）。 ▪ 置为 1 表示存在不同。 ▪ 位[14]：表示发生 EEPROM 写错误： ▪ 置为 1 表示检测到错误，	S 和 U
%SW97	保存/恢复功能的命令或诊断	当保存存储字时，该值表示被保存到内部 EEPROM 中的%MW 的字数。当恢复存储字时，该值被恢复到 RAM 的存储字数更新。对于保存操作，当数是 0 时，存储字将不被保存。用户必须用户逻辑程序定义。否则，该程序在控制器应用中将被置为 0，除了下述情况： 冷启动时，该字置为-1表示内部Flash EEPROM 没有被保存的存储字**%MW**文件。若冷启动时内部 Flash EEPROM包含一个存储字**%MW**文件，文件中被保存的存储字数的值必须在这个系统字**%SW97**中设置。	U

系统字	功能	描述	控制
%SW111	远程连接状态	表示：位 0 对应远程控制器 1，位 1 对应远程控制器 2，等等。 位[0]到[6]： ▌ 置为 0＝远程控制器 1-7 不存在 ▌ 置为 1＝远程控制器 1-7 存在 位[8]到[14]： ▌ 置为 0＝远程控制器 1-7 检测到远程 I/O ▌ 置为 1＝远程控制器 1-7 检测到扩展控制器	S
%SW112	远程连接配置/操作错误代码	00: 操作成功 01: 检测到时间超出（从机） 02: 检测到校验和出错（从机） 03: 配置不匹配（从机） 它由系统置为1且必须由用户复位。	S
%SW113	远程连接配置	表示：位0对应远程控制器1，位1对应远程控制器 2，等等。 位[0]到[6]： ▌ 置为 0＝远程控制器 1-7 没有被配置 ▌ 置为 1＝远程控制器 1-7 已被配置 位[8]到[14]： ▌ 置为 0＝远程 I/O 配置成远程控制器 1-7 ▌ 置为 1＝对等控制器配置成远程控制器 1-7	S
%SW114	调度模块使能	使或不使用户程序或操作显示可以执行调度模块操作。 位0：1 = 调度模块#0使能 ... 位15：1 = 调度模块#15使能 所有调度模块初始为使能状态。 如果调度模块被配置其默认值是FFFF 如果没有调度模块被配置其默认值是0	S 和 U
%SW118	基本控制器状态字	显示主控制器上检测到的故障。 位 9：0 = 外部故障或通信故障 位 12：0 = 没有安装 RTC 位13：0 = 配置故障（I/O扩展模块已配置但不存在或出现故障） 该字其它位均置为1且被保留。对一个没有故障的控制器，该字的值是FFFFh。	S
%SW120	扩展 I/O 模块正常	每个模块一位。 地址0 = 位 0 Address 0 = Bit 0 1 = 不正常 0 = 正常	S

表缩写描述　　缩写表：

缩写	描述
S	被系统控制
U	被用户控制

!

% 前缀，表示控制器中内部存储器地址，用于存储程序变量值，常量值，I/O 值，等等。

A

地址 控制器的内部寄存器，用于存储程序变量值，常量值，I/O 值，等等。地址用一个百分符号(%)前缀来标识。例如，%I0.1 表示控制器 RAM 存储器中的一个地址，包含输入通道 1 的值。

模拟电位计 一个应用的电压，可由应用程序调节和转换到数字值。

分析程序 一个命令，编译程序和检查程序错误：语法和结构错误，符号与地址不对应，程序使用资源不存在，以及程序与可用控制器存储器不适合。错误被显示在程序错误浏览器中。

动态数据表 在语言或操作屏幕中创建的表。当 PC 连接到控制器时，提供控制器变量查看并允许调试时强制值。可以保存为一个扩展名为.tat 的单独文件。

动态数据表编辑器	一个TwidoSoft应用程序中的专用窗口，用于查看和创建活动表。
应用程序	TwidoSoft 应用程序由程序，配置数据，符号，和文档组成。
应用程序浏览器	TwidoSoft 的一个专用窗口，显示一个应用程序的树形图。为应用程序的配置和查看提供方便。
应用程序文件	Twido 应用程序以文件类型.twd 存储。
ASCII	美国信息交换标准码。一种通信协议，使用七位表示文字数字式字符，包括字母，数字，和一些图形及控制字符。
行自动验证	当插入或修改列表指令，这个可选设置允许程序行在输入错误或未定符号时得到验证。您在退出该行时必须更正每个元素。选择使用参数对话框。
自动装载	一种特性，一般情况下均可用，在应用程序丢失或被破坏的情况下提供应用程序从备份卡到控制器 RAM 的自动传送。上电时，控制器将目前控制器 RAM 中的应用程序与可选备份存储卡（如果安装的话）中的应用程序进行比较。如果存在不同，则备份卡中的备份复制到控制器和内部 EEPROM。如果没有安装备份卡，则内部 EEPROM 中的应用程序复制到控制器。

B

备份	一个命令，复制控制器 RAM 中的应用程序到控制器内部 EEPROM 和可选备份存储卡（如果安装的话）。

C

线圈	一个梯形图元素，表示控制器的一个输出。

冷启动或重启 控制器启动，所有数据初始化为默认值，且程序开始运行所有变量被清空。所有的软件和硬件被初始化。装载一个新的应用程序到控制器RAM可能导致冷重启。任何没有备份电池的控制器一般上电时为冷启动。

注释行 在列表程序中，输入的注释可以与指令不同行。注释行不含有行号，且必须插入到圆括号和星号内，例如：(*COMMENTS GO HERE*)。

注释 注释是您输入到说明程序目的的文本。对梯形图程序，在梯级头输入最多三行文本描述梯级的目的。每行由 1 到 64 个字符组成。对列表程序，在未编号的程序行输入文本。必须插入到圆括号和星号内，例如：(*COMMENTS GO HERE*)。

一体型控制器 Twido 控制器类型，提供一个简单，一体化配置，和有限的扩展。模块型是 Twido 控制器的其它类型。

配置编辑器 专用 TwidoSoft 窗口，用于管理硬件和软件配置。

常量 一个配置值，程序执行时不能修改。

触点 一个梯形图元素，表示控制器的一个输入。

计数器 一个功能模块，用于事件计数（加或减计数）。

参照值浏览器 TwidoSoft 应用程序中的一个专用窗口，用于查看参照值。

参照值 一类用于应用程序的操作数，符号，行/梯级号，和运算符列表，用于简化应用程序的创建和管理。

D

数据变量 见变量。

日期/时钟功能 允许控制月，日，和一天的时间这些事件。见调度模块。

鼓型控制器 一个功能模块，其工作类似于机电式鼓形控制器，步的改变与外部事件有关。

E

EEPROM 电可擦除可编程只读存储器。Twido 有一个内部 EEPROM 和一个可选外部 EEPROM 存储卡。

擦除 此命令删除控制器中的应用程序，有两个选项：

- 删除控制器RAM，控制器内部EEPROM，和安装的可选备份卡的内容。
- 只删除安装的可选备份卡的内容。

可执行装载 一个 32 位 Windows 应用程序，用于下载一个新的固件可执行程序到一个 Twido 控制器。

扩展总线 扩展 I/O 模块使用这些总线连接到基本控制器。

扩展 I/O 模块 可选扩展 I/O 模块可用于增加 I/O 点到 Twido 控制器。（不是所有控制器模块都允许扩展。）

F

高速计数器 一个功能模块，提供比计数器功能模块更快的加/减计数。一个快速计数器计数最高频率可达 5 KHz。

FIFO 先入先出。一个功能模块，用于队列错作。

固件执行者 固件执行者是操作系统，执行您的应用程序和管理控制器工作。

强制 有意图地设置控制器输入和输出到值 0 或 1，即使和实际值不同。在激活程序时用于调试。

功能模块 一个程序体，输入和变量基于一个定义的功能如定时器或计数器被组织起来计算输出值。

G

Grafcet Grafcet 用于结构和图形格式的连续操作功能。
它是一个分析方法，将连续控制系统分解为一系列的步，以及有关的动作，转换，和条件。

I

初始状态 当 TwidoSoft 开始或还未打开应用程序时 TwidoSoft 的工作状态，显示在状态栏。

初始化 一个命令，设置所有数据值到初始状态。控制器必须处于停止或出错模式。

示例 程序中的一个独特对象，属于功能模块的一个特定类型。例如，在定时器格式%TMi，i 是一个编号表示示例。

指令列表语言 用指令列表语言(IL)编写的程序，由一系列被控制器顺序执行的指令组成。每个指令由行号，指令码，和操作数组成。

L

梯形图编辑器 专用 TwidoSoft 窗口，用于编辑一个梯形图程序。

梯形图语言 用梯形图语言编写的程序，由控制器程序指令的图形表示组成，包括控制器顺序执行的一系列梯级的触点，线圈和模块符号。

梯形图列表梯级 列表程序不能转化为梯形图语言的显示部分。

输入闭锁 到达的脉冲被捕获且被记录，以便应用程序以后检查。

LIFO 后入先出。一个功能模块，用于堆栈操作。

列表编辑器	简单的程序编辑器，用于创建和编辑列表程序。

M

主控制器	远程连接网络中配置成主机的 Twido 控制器。
存储卡	可选的备份存储卡，可以用于备份和存储应用程序（程序和配置数据）。存在两种容量：32 和 64 Kb。
存储器使用指示器	TwidoSoft 主窗口中状态栏的一部分，显示应用程序使用总的控制器存储器的百分比。当存储容量不足时提供警告。
Modbus	一个主-从通信协议，允许一个主机请求多个从机响应。
模块型控制器	Twido 控制器类型，提供扩展能力的灵活配置。一体型是 Twido 控制器的其它类型。
监视器状态	当一个 PC 以非写模式连到一个控制器时 TwidoSoft 的工作状态，显示在状态栏。

N

网格	一个梯级，位于网格中两个电势条之间，且由水平和垂直互相连接在一起的一组图形元素组成。一个梯级的最大范围是七行十一列。

O

离线操作	TwidoSoft 的一种工作模式，此时 PC 不与控制器连接且 PC 存储器中的应用程序与控制器存储器中的应用程序不同。您在离线操作情况下创建和开发应用程序。

离线状态 当 PC 与控制器不相连接时 TwidoSoft 的工作状态，显示在状态栏。

在线操作 TwidoSoft 的一种工作模式，此时 PC 与控制器连接且 PC 存储器中的应用程序与控制器存储器中的应用程序相同。在线操作可用于调试应用程序。

在线状态 当一个 PC 连接到一个控制器时 TwidoSoft 的工作状态，显示在状态栏。

操作数 一个数，地址，或符号，表示程序在指令中可以操作的一个值。

工作状态 表示 TwidoSoft 状态。显示在状态栏。有四个工作状态：初始，离线，在线，和监控状态。

运算符 一个符号或代码，指定一个操作由指令完成。

P

PC 个人计算机。

对等控制器 一个 Twido 控制器，在远程连接网络中配置为从机。应用程序可在对等控制器存储器中执行且程序可以访问本地和扩展 I/O 数据，但是 I/O 数据不能传递给主控制器。对等控制器的运行程序通过网络字(%INW 和 %QNW)传递信息给主控制器。

PLC Twido 可编程控制器。有两类控制器：一体型和模块型。

PLS 脉冲发生器。一个功能模块，生成一个方波，50%开和 50%闭的占空比。

参数选择 一个对话框，有选择项设置列表和梯形图程序编辑器。

程序错误浏览器 专用 TwidoSoft 窗口，用于查看程序错误和警告。

可编程控制器 Twido 控制器。有两类控制器：一体型和模块型。

保护 请参考两种不同的应用程序保护：密码保护和控制器应用程序保护，前者提供访问控制，后者禁止应用程序的所有读和写。

PWM 脉宽调制。一个功能模块，生成一个矩形波，其占空比可通过程序设置来改变。

R

RAM 随机存储器。Twido 应用程序被下载到内部可变 RAM 执行。

实时时钟 一个选件，即便控制器在有限时间没有上电也可以保持时间。

映像输出 在计数模式，超高速计数器的当前值(%VFC.V)通过与配置阈值比较测量以决定这些专用输出的状态。

寄存器 控制器内部专用寄存器，专用于 LIFO/FIFO 功能模块。

远程控制器 一个 Twido 控制器，在远程连接网络中配置与主控制器通信。

远程连接 高速主/从总线，用于主控制器和最多七个远程控制器（从机）之间的少量数据通信。可配置两种远程控制器传送数据到主控制器：对等控制器传送应用程序数据，或者远程 I/O 控制器传送 I/O 数据。远程连接网络可由这两种类型混合组成。

资源管理器 TwidoSoft 的一个部分，监控应用程序在通过跟踪应用程序软件对象的参数进行编程和配置时的存储器需求。考虑一个对象在应用程序中被引用时是否是在列表指令或梯级中用作一个操作数。显示有关总的存储器被使用的百分比状态信息，以及在存储容量不足时提供警告。见存储器使用指示器。

可逆指令 一个编程方法，允许指令可被交替查看，或者是列表指令，或者是梯级。

RTC 见实时时钟。

RTU 远程终端单元。协议使用八位用于控制器和 PC 间的通信。

运行 一个命令，使控制器运行应用程序。

梯级头 位于梯级上方的一个面板，用于说明梯级的目的。

S

扫描 控制器扫描一个程序并本质上执行三个基本功能。首先，它读取输入并将这些值放到存储器。其次，它一次执行应用程序的一条指令并存储结果到存储器。最后，它使用这些结果更新输出。

扫描模式 指定控制器怎样扫描一个程序。有两种扫描模式：常规（循环的），控制器连续扫描，或周期的，控制器的扫描在另一个扫描开始之前持续一段选择的时间（范围 2 – 150 毫秒）。

调度模块 一个功能模块，用于编程日期和时间功能以控制事件。需要实时时钟选件。

步 Grafcet 步指定了自动顺序操作的状态。

停止 一个命令，使控制器停止运行一个应用程序。

符号 一个符号是一个串，最多包含 32 个文字数字式字符，且第一个字符是字母。它允许您个性化控制器对象以方便应用程序维护。

符号表 符号表用于应用程序。显示在符号编辑器中。

T

阈值输出 一类线圈，由超高速计数器(%VFC)根据配置设置直接控制。

定时器 一个功能模块，用于选择一个时间周期控制事件。

Twido Schneider Electric 控制器系列，包含两类控制器（一体型和模块型），扩展模块增加 I/O 点，以及一些选件如实时时钟，通信卡，操作显示模块，和备份存储卡。

TwidoSoft 一个 32 位 Windows，图形开发软件，用于 Twido 控制器配置和编程。

U

未定义符号 符号没有变量地址。

V

变量 存储器单元，可由程序寻址和修改。

超高速计数器 一个功能模块，提供比计数器和高速计数器功能模块更快的计数。超高速计数器计数频率最高可达 20 KHz。

W

热重启 控制器在电源损失后上电，不改变应用程序。控制器恢复到电源损失前的状态并完成处理中的扫描。所有应用程序数据被保存。这个特性仅为模块型控制器所有。