SoMachine Basic

通用功能库指南

04/2014

EIO0000001479.01

www.schneider-electric.com

Schneider Electric

本文档中提供的信息包含有关此处所涉及产品之性能的一般说明和 / 或技术特性。本文档并非用于（也不代替）确定这些产品对于特定用户应用场合的适用性或可靠性。任何此类用户或集成者都有责任就相关特定应用场合或使用方面对产品执行适当且完整的风险分析、评估和测试。 Schneider Electric 或其任何附属机构或子公司对于误用此处包含的信息而产生的后果概不负责。如果您有关于改进或更正此出版物的任何建议，或者从中发现错误，请通知我们。

在安装和使用本产品时，必须遵守国家、地区和当地的所有相关的安全法规。出于安全方面的考虑和为了帮助确保符合归档的系统数据，只有制造商才能对各个组件进行维修。

当设备用于具有技术安全要求的应用场合时，必须遵守有关的使用说明。

未能使用 Schneider Electric 软件或认可的软件配合我们的硬件，则可能导致人身伤害、损害或不正确的操作结果。

不遵守此信息可能导致人身伤害或设备损坏。

目录

安全信息

重要信息

声明

在尝试安装、操作或维护设备之前，请仔细阅读下述说明并通过查看来熟悉设备。下述特别信息可能会在本文其他地方或设备上出现，提示用户潜在的危险，或者提醒注意有关阐明或简化某一过程的信息。

在“危险”标签上添加此符号表示存在触电危险，如果不遵守使用说明，会导致人身伤害。

这是提醒注意安全的符号。提醒用户可能存在人身伤害的危险。请遵守所有带此符号的安全注意事项，以避免可能的人身伤害甚至死亡。

⚠ 危险

“危险”表示极可能存在危险，如果不遵守说明，可导致严重的人身伤害甚至死亡。

⚠ 警告

“警告”表示可能存在危险，如果不遵守说明，可导致严重的人身伤害甚至死亡，或设备损坏。

⚠ 注意

“注意”表示可能存在危险，如果不遵守说明，可导致严重的人身伤害或设备损坏。

注意

“注意”用于表示与人身伤害无关的危害。

请注意

电气设备的安装、操作、维修和维护工作仅限于合格人员执行。 Schneider Electric 不承担由于使用本资料所引起的任何后果。

专业人员是指掌握与电气设备的制造和操作及其安装相关的技能和知识的人员，他们经过安全培训能够发现和避免相关的危险。

关于本书

概览

文档范围

本节介绍如何使用通过 SoMachine Basic 软件所创建程序中的功能块和指令。其中，描述的内容适用于 SoMachine Basic 所支持的所有 Logic Controller。

有效性说明

本手册中的信息仅适用于兼容 SoMachine Basic 的产品。

本文档已随 SoMachine Basic V1.1 的发布进行了更新。

本文档中描述的设备技术特性在网站上也有提供。要在线访问此信息：

步骤	操作
1	访问 Schneider Electric 主页 *www.schneider-electric.com*。
2	在 **Search** 框中键入产品参考号或产品系列名称。 ● 型号 / 产品系列中不得包括空格。 ● 要获得有关类似模块分组的信息，请使用星号 (*)。
3	如果您输入参考号，则转到 **Product datasheets** 搜索结果，单击您感兴趣的参考号。 如果您输入产品系列的名称，则转到 **Product Ranges** 搜索结果，单击您感兴趣的产品系列。
4	如果 **Products** 搜索结果中出现多个参考号，请单击您感兴趣的参考号。
5	根据屏幕大小，您可能需要向下滚动查看数据表。
6	要将数据表保存为 .pdf 文件或打印数据表，请单击 **Download XXX product datasheet**。

本手册中介绍的特性应该与在线显示的那些特性相同。依据我们的持续改进政策，我们将不断修订内容，使其更加清楚明了，更加准确。如果您发现手册和在线信息之间存在差异，请以在线信息为准。

相关的文件

文件名称	参考编号
SoMachine Basic 操作指南	EIO0000001354（英语） EIO0000001355（法语） EIO0000001356（德语） EIO0000001357（西班牙语） EIO0000001358（意大利语） EIO0000001359（简体中文） EIO0000001366（波兰语） EIO0000001367（土耳其语）

您可以从我们的网站下载这些技术出版物和其它技术信息，网址是：www.schneider-electric.com。

关于产品的资讯

⚠ 警告

失去控制

- 任何控制方案的设计者都必须考虑到控制路径可能出现故障的情况，并为某些关键控制功能提供一种方法，使其在出现路径故障时以及出现路径故障后恢复至安全状态。这些关键控制功能包括紧急停止、越程停止、断电重启以及类似的安全措施。
- 对于关键控制功能，必须提供单独或冗余的控制路径。
- 系统控制路径可包括通讯链路。必须对暗含的无法预料的传输延迟或链路失效问题加以考虑。
- 遵守所有事故预防规定和当地的安全指南。[1]
- 为了保证正确运行，在投入使用前，必须对设备的每次执行情况分别进行全面测试。

不遵循上述说明可能导致人员伤亡或设备损坏。

[1] 有关详细信息，请参阅 NEMA ICS 1.1（最新版）中的 “ 安全指导原则 - 固态控制器的应用、安装和维护 ” 以及 NEMA ICS 7.1 （最新版）中的 “ 结构安全标准及可调速驱动系统的选择、安装与操作指南 ” 或您特定地区的类似规定。

⚠ 警告

意外的设备操作

- 仅使用 Schneider Electric 认可的可与本设备配合使用的软件。
- 每次更改物理硬件配置后，请更新应用程序。

不遵循上述说明可能导致人员伤亡或设备损坏。

章 1
简介

概述

本章提供了如何使用源代码示例和块的相关信息，这些块需要运行本文档中给出的许多运算和赋值指令的示例。

本章包含了哪些内容？

本章包含了以下主题：

如何使用源代码示例

概述

除非明确提出，否则本手册中包含的源代码示例对梯形图和指令列表编程语言均有效。整个示例可能需要多个梯级。

可转换性过程

本手册中仅显示指令列表源代码。

要获得对应的梯形图源代码，请执行以下操作：

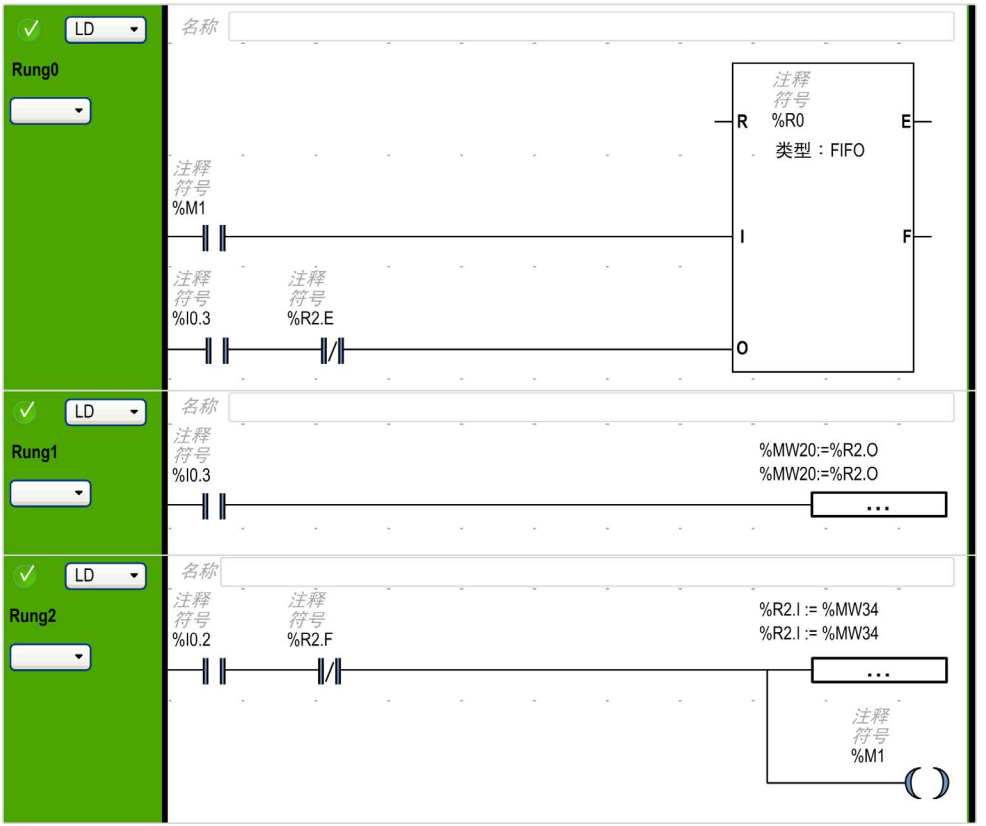
步骤	操作
1	在 SoMachine Basic 中，创建一个包含空梯级的新 POU。
2	在此梯级中，单击 **LD > IL** 按钮以显示指令列表源代码。
3	选择并复制 (**Ctrl+C**) 示例程序的第一个梯级的源代码。
4	右键单击第一个指令的行号 **0000**，然后选择**粘贴指令**，将源代码粘贴到梯级：  **注意：**如果通过在缺省 LD 操作符之前插入行来粘贴指令，那么务必从梯级的最后一行中删除 **LD** 指令。
5	单击 **IL > LD** 按钮以显示梯形图源代码。
6	对示例程序中的任何其他梯级重复上述步骤。单击工具栏上的  以添加新梯级。

示例

指令列表程序：

梯级	源代码
0	BLK %R0 LD %M1 I LD %I0.3 ANDN %R2.E O END_BLK
1	LD %I0.3 [%MW20:=%R2.O]
2	LD %I0.2 ANDN %R2.F [%R2.I:=%MW34] ST %M1

相应的梯形图：


LD
Rung0
名称
注释
符号
R
%R0
E
类型：FIFO
注释
符号
%M1
I
F
注释
符号
%I0.3
注释
符号
%R2.E
O
LD
Rung1
名称
注释
符号
%I0.3
%MW20:=%R2.O
%MW20:=%R2.O
...
LD
Rung2
名称
注释
符号
%I0.2
注释
符号
%R2.F
%R2.I := %MW34
%R2.I := %MW34
...
注释
符号
%M1

操作块

将 IL 运算和赋值指令插入梯形图

您可以使用**操作块**图形符号将指令列表运算和赋值指令插入梯形图梯级：

要将操作块插入梯级，请执行以下操作：

步骤	操作
1	单击工具栏上的**操作块**  按钮。
2	单击要插入**操作块**的梯级的操作区 （最后 2 列）。
3	双击**运算表达式**行。
4	输入有效的指令列表运算或赋值指令。

获取语法帮助

如果指令列表运算或赋值指令的语法不正确，则**运算表达式**框的边框会变为红色。要获取帮助，请执行以下操作之一：

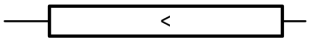
- 将鼠标移到**运算表达式**行上 , 或
- 选择**工具 →程序消息**。

比较块

将 IL 比较表达式插入梯形图

您可以使用**比较块**图形符号将指令列表比较表达式插入梯形图梯级：

符号
比较表达式



继续执行下列步骤：

步骤	操作
1	单击工具栏上的**比较块**  按钮。
2	单击梯级中的任意位置以插入**比较块**。
3	双击**比较表达式**行。
4	输入有效的指令列表比较运算。

获取语法帮助

如果指令列表比较运算的语法不正确，则**比较表达式**框的边框会变为红色。要获取帮助，请执行以下操作之一：

- 将鼠标移到**比较表达式**行上，或
- 选择**工具** →**程序消息**。

章 2
语言对象

本章包含了哪些内容？

本章包含了以下主题：

对象

概述

在 SoMachine Basic 中，*对象*一词用于表示保留供应用程序使用的 Logic Controller 存储器的区域。对象可能是：

- 简单软件变量，例如存储器位和字
- 数字量或模拟量输入和输出的地址
- 控制器内部变量，例如系统字和系统位
- 预定义的系统功能或功能块，例如定时器和计数器。

将为某些对象类型预先分配控制器存储器，或者在应用程序下载到 Logic Controller 时自动分配。

分配存储器后，对象只能由程序进行寻址。使用前缀 % 对对象进行寻址。例如， %MW12 是存储器字的地址， %Q0.3 是嵌入式数字量输出的地址，而 %TM0 是 Timer 功能块的地址。

存储器位对象

简介

存储器位对象是位类型的软件变量，可以用作操作数并可以由布尔指令测试。

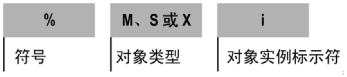
位对象示例：

- 存储器位
- 系统位
- 步进位
- 从字中提取的位

有效对象的范围为 0 到应用中所用的最大引用 （参见逻辑控制器的*编程指南*）。

语法

使用以下格式对内存位对象、系统位对象和步进位对象进行寻址：



下表介绍了寻址格式中的元素：

分组	项	描述
符号	%	百分号始终出现在软件变量之前。
对象类型	M	程序正在运行时，内存位存储中间值。
	S	系统位为控制器提供状态和控制信息。
	X	步骤位提供 Grafcet 步骤活动的状态。
对象实例标示符	i	对象的标识符，表示对象在存储器中的顺序实例。对象的最大数量取决于配置为可用存储器的限值的对象容量。有关可用存储器的最大数量，请参见您的逻辑控制器的*编程指南*。

有关 I/O 位寻址的信息，请参见 I/O 对象 (参见第 *23* 页)。

有关从字中抽取的位的寻址信息，请参见从字对象中抽取位 (参见第 *28* 页)。

描述

下表列出并介绍了可用作布尔指令中操作数的内存位对象、系统位对象和步骤位对象。

类型	描述	地址或值	写访问 (1)
立即值	0 或 1 （False 或 True）	0 或 1	–
存储器	存储器位是用于存储二进制值的内部存储器区域。 **注意：**未使用的 I/O 对象不得用作内存位。	%Mi	是
系统	通过系统位 %S0 到 %S127，您可以监视控制器的运作是否正确，应用程序的运行是否合适，并控制某些系统级功能。	%Si	取决于 i
Grafcet 步骤	位 %X1 到 %Xi 与 Grafcet 步骤相关联。当相应步骤处于活动状态时，将步骤位 Xi 设置为 1；而当相应步骤处于禁止状态时，将步骤位设置为 0。	%Xi	是
(1) 由程序或使用动态数据表写入。			

示例

下表显示了位对象寻址的部分示例：

位对象	描述
%M25	内存位数 25
%S20	系统位编号 20
%X4	Grafcet 阶跃数 4

I/O 对象

简介

I/O 对象同时包括位和字。每个物理输入和输出均映射到内部存储器中的相应对象。 I/O 位对象可用作操作数，并且可以由布尔值指令测试。 I/O 字对象可在大部分非布尔指令中使用，如包含算术操作符的函数和指令。

I/O 对象的示例：

- 数字量输入
- 数字量输出
- 模拟量输入
- 模拟量输出
- 通信输入和输出

有效对象的范围是从 0 到您的控制器配置且支持的最大值 （请参见您的逻辑控制器的硬件指南和编程指南）。

语法

下图显示输入 / 输出地址格式：

下表介绍了寻址格式的组成部分：

组件	项	值	描述
符号	%	–	百分号始终位于内部地址之前。
对象类型	I	–	数字输入 （位对象）
	Q	–	数字输出 （位对象）
	IW	–	模拟量输入值 （字对象）
	QW	–	模拟量输出值 （字对象）
	IWS	–	模拟量输入状态 （字对象）
	QWS	–	模拟量输出状态 （字对象）
模块编号	y	0	Logic Controller 上的嵌入式 I/O 通道。
		1...*m*(1)	直接连接到控制器的扩展模块上的 I/O 通道。
		m+1...*n*(2)	使用 TM3 发射器 / 接收器模块连接的扩展模块上的 I/O 通道。
通道编号	z	0...31	Logic Controller 或扩展模块上的 I/O 通道编号。可用通道的编号取决于 Logic Controller 型号或扩展模块类型。
(1) *m* 是所配置的本地模块的数量 （最多为 7）。 **(2)** *n* 是所配置的远程模块的数量 （最多为 n+7）。最大位置编号为 14。			

描述

下表列出并介绍了指令中用作操作数的所有 I/O 对象：

类型	地址或值	写访问 (1)	描述
输入位	%Iy.z(2)	否 (3)	这些位为物理数字 I/O 电气状态的“逻辑映像”。它们存储在数据存储器中并在每次扫描程序逻辑之间更新。
输出位	%Qy.z(2)	是	
输入字	%IWy.z(2)	否	这些字对象包含对应通道的模拟量值。
输出字	%QWy.z(2)	是	
输入字状态	%IWSy.z(2)	否	这些字对象包含对应模拟量通道的状态。
输出字状态	%QWSy.z(2)	否	
(1) 由程序或使用动态数据表写入。 **(2)** y 是模块数， z 是通道数。有关 y 和 z 的说明，请参见 I/O 寻址语法 (参见第 *23* 页)。 **(3)** 尽管无法写入到输入位，但可以强制输入位。			

示例

下表显示了一些 I/O 寻址的示例：

I/O 对象	描述
%I0.5	控制器上的数字量输入通道编号 0 （嵌入式 I/O 为模块编号 0）。
%Q3.4	地址 3 处的扩展模块上的数字量输出通道编号 4 （扩展模块 I/O）。
%IW0.1	控制器上的模拟量输入 1 （嵌入式 I/O）。
%QW2.1	地址 2 处的扩展模块上的模拟量输出 1 （扩展模块 I/O）。
%IWS0.1	控制器上的模拟量输入 1 的模拟量输入的状态 （嵌入式 I/O）。
%QWS1.1	地址 1 处扩展模块上模拟量输出 1 的模拟量输出状态 （扩展模块 I/O）。

字对象

简介

字对象存储在数据存储器中，以 16 位字形式编址，可以包含介于 -32768 到 32767 之间的整数值（**高速计数器**功能块除外，其值介于 0 到 65535 之间）。

字对象示例：

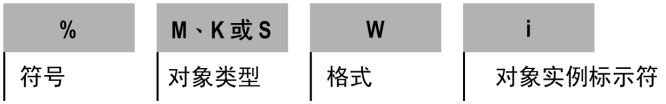
- 立即值
- 内存字 (%MWi)
- 常量字 (%KWi)
- I/O 交换字 （%IWi、 %QWi、 %IWSi、 %QWSi）
- 系统字 (%SWi)
- 功能块 （配置和 / 或运行时数据）

有效对象的范围从 0 到应用程序中所用的最大引用 （请参见 Logic Controller 的 《编程指南》）。

例如，若存储器字应用程序中的最大引用为 %MW9，则 %MW0 到 %MW9 为已分配的空间。本例中的 %MW10 无效，且无法从内部或外部进行访问。

语法

使用下列格式对内存字、常量字和系统字进行寻址：



下表介绍了寻址格式中的元素：

分组	项	描述
符号	%	百分号始终位于内部地址之前。
对象类型	M	程序正在运行时，内存字存储值。
	K	常量字存储常量值或字母数字消息。其内容只能用 SoMachine Basic 进行写入或修改。
	S	系统字为 Logic Controller 提供状态和控制信息。
格式	W	16 位字。
对象实例标示符	i	对象的标识符，表示对象在存储器中的顺序实例。对象的最大数量取决于配置为可用存储器的限值的对象容量。有关可用存储器的最大数量，请参见您的逻辑控制器的*编程指南*。

格式

字或值的内容以 16 位二进制代码 （二进制补码）形式存储在用户存储器中，并遵从下列约定：

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	位位置
±	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	位值

根据惯例，在有符号二进制表示法中，将位 15 分配为编码值的符号位：

- 位 15 设为 0：字的内容为正值。
- 位 15 设为 1：字的内容为负值 （负值以二进制补码逻辑表示）。

字和立即值 （参见适用于无符号整数的异常列表 (参见第 *26* 页)）可采用以下格式进行输入或检索：

- 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）
  替代语法：#A536
- ASCII 格式规则如下：
  - 该函数始终首先读取最高有效字节。
  - 不在 [0 - 9] ([16#30 - 16#39]) 范围内的 ASCII 字符均可用作结束字符，但减号 “-”(16#2D) 作为首个字符时除外。
  - 在溢出 （>32767 或 <-32768）情况下，系统位 %S18 （算术溢出或检测到的错误）将设为 1，并返回值 32767 或 -32768。
  - 如果操作数的第一个字符是 “end” 字符，则返回值 0，且位 %S18 将设为 1。

  例如， "HELLO"：
  - %MW0:="HE"
  - %MW1:="LL"
  - %MW2:="O"

异常列表

下表列出了无符号整数的对象的值范围：

对象	值
%SW	0...65535
%FC.V 和 %FC.P	0...65535
%FC.VD 和 %FC.PD	0...4294967295
%HSC.V、 %HSC.P、 %HSC.S0、 %HSC.S1 和 %HSC.C	0...65535
%HSC.DV、 %HSC.PD、 %HSC.S0D、 %HSC.S1D 和 %HSC.CD	0...4294967295
%HSC.T	100...1000
%PWM.P	0...32767

对象	值
%PWM.R	0...100
%PLS.P	0...32767
%PLS.N	0...32767
%PLS.ND	0...2147483647

除了异常列表中的对象外，其他所有数据都有下列值范围：
- 字：-32768...32767
- 双字：-2147483648...2147483647

描述

下表描述字对象：

字	描述	地址或值	写访问 (1)
立即值	这些整数值与 16 位字的格式相同，因此可将值分配给这些字。	-	否
	基数 10 （十进制）	-32768 到 32767	
	基数 16 （十六进制）	16#0000 到 16#FFFF	
存储器	在数据存储器中操作期间用作 “ 工作 ” 字来存储值。	%MWi	是
常量	存储常量或字母数字消息。其内容仅能在配置时用软件进行写入或修改。	%KWi	否
系统	这些 16 位字具有多个功能： ● 通过读取 %SWi 字访问直接来自控制器的数据。 ● 在应用程序上执行操作 （例如，调整计划块）。	%SWi	取决于 i
功能块	这些字对应于功能块的当前参数或值。	%TM2.P 和 %Ci.P 等。	是
(1) 由程序或使用动态数据表写入。			

可用对象的最大值由逻辑控制器决定。有关对象的最大值，请参见逻辑控制器的*编程指南*。

示例

下表显示了字对象寻址的部分示例：

字对象	描述
%MW15	内存字编号 15
%KW26	常量字编号 26
%SW30	系统字编号 30

从字对象中抽取位

下表介绍了如何从以下字对象中抽取 1 个 16 位：

字对象	地址或值	写访问 (1)
存储器	%MWi:Xk	是
系统	%SWi:Xk	取决于 i
常量	%KWi:Xk	否
输入值	%IWy.z:Xk(2)	否
输出值	%QWy.z:Xk(2)	是
输入状态	%IWSy.z:Xk(2)	否
输出状态	%QWSy.z:Xk(2)	是

(1) 由程序或使用动态数据表写入。
(2) 有关 I/O 字对象的信息，请参见对 I/O 对象寻址 (参见第 *23* 页)。
Xk 表示必须从字对象中抽取的位数。例如， %MW0.X3 ；将抽取内存字 %MW0 中第三顺位保存的位。

浮点数和双字对象

简介

浮点对象是实际值；即，包含分数部分的值 （例如， 3.4E+38、 2.3 或 1.0）。

双字由存储于数据存储器中的四个字节组成，且包含一个介于 --2147483648 和 +2147483647 之间的 2 的补码值。

所有 Logic Controller 均不支持浮点数和双字运算。

有关兼容性，请参阅逻辑控制器的*编程指南*

浮点数格式和值

所用浮点数格式为标准 IEEE STD 734-1985 （等同于 IEC 559）。字长为 32 位，对应于单小数点浮点数。

下表显示了浮点值的格式：

位 31	位 {30...23}	位 {22...0}
指数符号	指数	有效位数

表示浮点数的显示精度为 2...24，此时不必显示六个以上的小数位。

注意：值 “1285 ” 将被视为整数值，要使其被识别为浮点值，则必须将其写为下列格式：1285.0

针对浮点数的算术函数的极限范围

下表介绍针对浮点数对象的算术函数的极限范围：

算术函数		**极限范围和无效运算**	
类型	**语法**	**#QNAN （无效）**	**#INF （无穷）**
操作数的平方根	`SQRT(x)`	x < 0	x > 1.7E38
整数的实数幂 `EXPT(%MF,%MW)`	`EXPT(y, x)` （其中： `x^y = %MW^%MF`)	x < 0	y.ln(x) > 88
以 10 为底数的对数	`LOG(x)`	x <= 0	x > 2.4E38
自然对数	`LN(x)`	x <= 0	x > 1.65E38
自然指数	`EXP(x)`	x < 0	x > 88.0

有效性检查

当结果不在有效范围内时，系统位 %S18 将设为 1。

状态字 %SW17 指示浮点运算中已检测到的错误的原因。

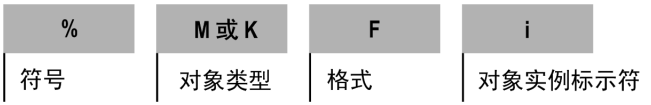
字 %SW17 的不同位：

%SW17:X0	无效运算，结果非数字 （1.#NAN 或 -1.#NAN）
%SW17:X1	保留
%SW17:X2	除 0，结果为无穷 （-1.#INF 或 1.#INF）
%SW17:X3	结果的绝对值大于 +3.402824e+38，结果为无穷 （-1.#INF 或 1.#INF）
%SW17:X4 to X15	保留

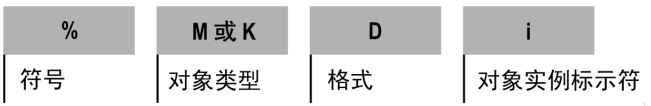
冷启动时系统会将该字复位为 0，程序也会因重复使用目的将其复位为 0。

语法

使用下列格式对内存浮点对象和常量浮点对象进行编址：



使用下列格式对内存双字对象和常量双字对象进行编址：



下表介绍了寻址格式中的元素：

分组	项	描述
符号	%	百分号始终位于内部地址之前。
对象类型	M	内存对象用于在程序运行时存储中间值。
	K	常量用于存储常数值或字母数字消息 （仅适用于双字）。
格式	F	32 位浮点对象。
	D	32 位双字对象。
对象实例标示符	i	表示内存中对象实例 （连续位置）的标示符。有关对象的最大值，请参见逻辑控制器的*编程指南*。

浮点对象和双字对象的描述

下表介绍浮点对象和双字对象：

对象类型	描述	地址	写访问
立即值	与 32 位对象具有相同格式的整数 （双字）或小数（浮点数）。	-	否
内存浮点	用于在操作过程中将值存储在数据存储器中的对象。	%MFi	是
内存双字		%MDi	是
浮点常量值	用于存储常量。	%KFi	是 （非通过程序）
双常量		%KDi	是 （非通过程序）

注意：最大对象数取决于 Logic Controller。有关详细信息，请参阅硬件平台的 《编程指南》。

示例

下表显示了浮点数对象和双字对象寻址的部分示例：

对象	描述
%MF15	编号为 15 的内存浮点对象
%KF26	编号为 26 的常量浮点对象
%MD15	编号为 15 的内存双字
%KD26	编号为 26 的常量双字

对象间重叠的可能性

单、双长度和浮点字存储在一个存储区中的数据空间内。因此，浮点字 %MFi 和双字 %MDi 对应于单长度字 %MWi 和 %MWi+1（字 %MWi 包含字 %MFi 的最低有效位，而字 %MWi+1 包含字 %MFi 的最高有效位）。

下表显示浮点内部字和双内存字的重叠方式：

浮点常量和双常量	奇地址	内存字
%MF0 / %MD0		%MW0
	%MF1 / %MD1	%MW1
%MF2 / %MD2		%MW2
	%MF3 / %MD3	%MW3
%MF4 / %MD4		%MW4
	...	%MW5
...		...
	%MFi / %MDi	%MWi
%MFi+1 / %MDi+1		%MWi+1

下表显示浮点常量和双常量的重叠方式：

浮点常量和双常量	奇地址	内存字
%KF0 / %KD0		%KW0
	%KF1 / %KD1	%KW1
%KF2 / %KD2		%KW2
	%KF3 / %KD3	%KW3
%KF4 / %KD4		%KW4
	...	%KW5
...		...
	%KFi / %KDi	%KWi
%KFi+1 / %KDi+1		%KWi+1

示例：

%MF0 对应于 %MW0 和 %MW1。 %KF543 对应于 %KW543 和 %KW544。

结构化对象

简介

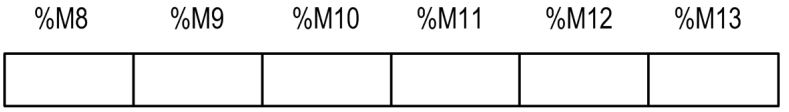
结构化对象是相邻对象的组合。 SoMachine Basic 支持以下类型的结构化对象：

- 位字符串
- 字表
- 双字表
- 浮点字表

位字符串

位字符串是一系列具有相同类型和已定义长度 (L) 的相邻对象位。位字符串从字节边界处开始引用。

示例：位字符串 `%M8:6`



注意：`%M8:6` 有效 （8 是 8 的倍数），而 `%M10:16` 无效 （10 不是 8 的倍数）。

位字符串可与赋值指令 (参见第 *47* 页) 搭配使用。

可用位类型

位字符串的可用位类型：

类型	地址	写访问
数字量输入位	`%I0.0:L` 或 `%I1.0:L` (1)	否
数字量输出位	`%Q0.0:L` 或 `%Q1.0:L` (1)	是
系统位	`%Si:L` i 是 8 的倍数	取决于 i
Grafcet 步进位	`%Xi:L` i 是 8 的倍数	是 （由程序确定）
存储器位	`%Mi:L` i 是 8 的倍数	是
(1) 仅 I/O 位 0 到 16 可在位字符串中读取。对于带 24 或 32 个 I/O 通道的 Logic Controller，无法在位字符串中对超过 16 的位进行读取。 **L** 表示结构对象 （位字符串、字表、双字表和浮点字表）的长度。		

位数取决于 Logic Controller。有关详细信息，请参阅硬件平台的 《编程指南》。

字表

字表是一系列具有相同类型和已定义长度 （L，最大值为 255）的相邻字。

示例：字表 %KW10:7

字表可与赋值指令 (参见第 *47* 页) 搭配使用。

可用字类型

字表的可用字类型：

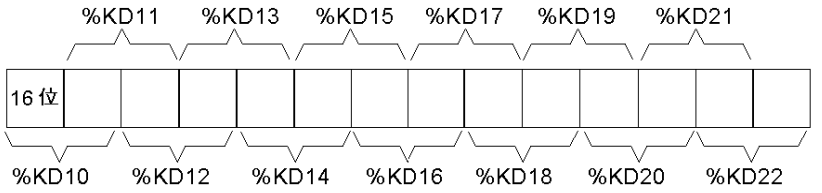
类型	地址	写访问
字存储器	%MWi:L	是
常量字	%KWi:L	否
系统字	%SWi:L	取决于 i

字数取决于 Logic Controller。有关详细信息，请参阅硬件平台的 《编程指南》。

双字表

双字表是一系列具有相同类型和已定义长度 （L，最大值为 255）的相邻字。

示例：双字表 %KD10:7



双字表可与赋值指令 (参见第 *47* 页) 搭配使用。

可用双字类型

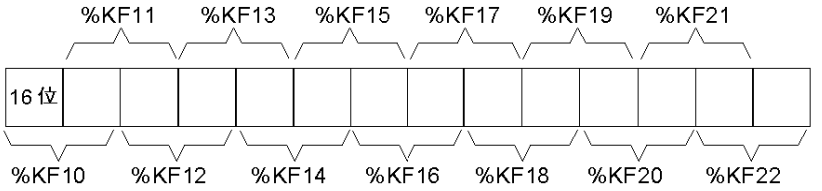
双字表的可用字类型：

类型	地址	写访问
字存储器	%MDi:L	是
常量字	%KDi:L	否

浮点字表

浮点字表是一系列具有相同类型和已定义长度 （L，最大值为 255）的相邻字。

示例：浮点字表 %KF10:7



浮点字表可与赋值指令 (参见第 *47* 页) 搭配使用。

可用浮点字类型

浮点字表的可用字类型：

类型	地址	写访问
字存储器	%MFi:L	是
常量字	%KFi:L	否

索引对象

简介

索引对象为带有索引对象地址的单字、双字或浮点对象。共有两种对象寻址方式：

- 直接寻址
- 变址寻址

直接寻址

对象的直接地址在编写程序时进行设置和定义。

示例： %M26 表示直接地址为 26 的内存位。

变址寻址

对象的索引地址提供了一种通过向对象的直接地址添加索引以修改对象地址的方法。索引的内容将添加到对象的直接地址中。索引由内存字 %MWi 定义。

示例： %MW108[%MW2] 表示所用地址由直接地址 108 与字 %MW2 的内容相加而成的字。

如果字 %MW2 的值为 12，则写入 %MW108[%MW2] 等同于写入 %MW120 （108 加 12）。

可进行变址寻址的对象

下表介绍适用于变址寻址的可用对象类型：

类型	地址	写访问
字存储器	%MWi[MWj]	是
常量字	%KWi[%MWj]	否
双字存储器	%MDi[MWj]	是
常量双字	%KDi[%MWj]	否
存储器浮点数	%MFi[MWj]	是
常量浮点数	%KFi[%MWj]	否
i 表示内存中对象实例 （顺序位置）的对象实例标示符。有关对象的最大值，请参见逻辑控制器的编程指南。 **j** 索引对象的对象实例标示符，该索引对象的内容已添加到其他对象的直接地址中。		

索引对象可与赋值指令 (参见第 *60* 页) 搭配使用，也可在比较指令 (参见第 *56* 页) 中使用。

此类寻址可通过使用程序修改索引对象内容的方式，使同一类型 （例如，内存字和常量）的一系列对象依次经过扫描。

索引溢出系统位 %S20

在索引对象的地址超过包含同类对象的存储器区限制时，将发生索引溢出。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应用程序中直接引用的最大字。

发生索引溢出时，系统位 %S20 将设为 1，并将索引值 0 分配给对象。

注意：用户负责监控任何溢出情况。此时，程序须读取 %S20 以便进行相关处理。随后，用户应确认已将其复位为 0。

%S20 （初始状态 = 0）：

- 发生索引溢出时：由控制器设为 1。
- 确认溢出：修改索引后在程序中手动设置为 0。

⚠ 警告

意外的设备操作

- 写入编辑指令，以测试用于数学运算的操作数是否有效。
- 避免在数学运算中使用不同数据类型的操作数。
- 始终监控分配的系统位以指示无效的数学结果。

不遵循上述说明可能导致人员伤亡或设备损坏。

功能块对象

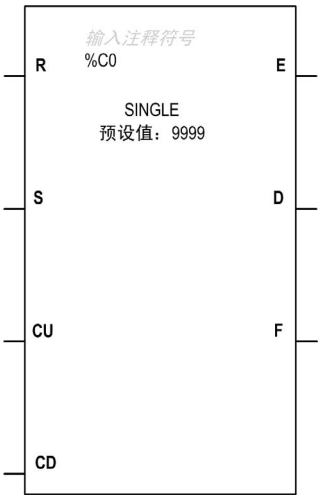
简介

功能块是一种可接受一个或多个输入值并返回一个或多个输出值的可重复使用对象。功能块始终通过实例（具有其专用名和变量的功能块）进行调用。每个功能块实例在从一个调用到另一调用时都会保持一种状态（输出和内部变量）。

注意：功能块（%FC、%HSC、%PLS 和 %PWM）和状态警报直接推动其输入和输出（%I0.x 和 %Q0.x，在配置中受到影响），而与控制器循环无关。控制器不会更新映像位（%I0.x 和 %Q0.x）。因此，这些输入和输出位不能直接在用户程序中使用，而且使用这些输入 / 输出的动态表不会显示这些输入 / 输出的当前状态。

示例

本示意图显示 StepCounter 功能块：



位对象

位对象对应于功能块输出。使用以下任何一种方法，布尔测试指令均可以访问这些位：

- 如果这些位与可逆编程 (参见第 *125* 页) 中的块连接，则采用“直接”法（例如，LD E）。
- 通过指定块类型（例如，LD %Ci.E）。

可以以指令形式访问输入。

字对象

字对象对应指定的参数和值，如下所述：

- 块配置参数：某些参数可由程序访问 （例如，预选参数），而另外一些参数则无法通过程序访问 （例如，时基）。
- 当前值：例如， %Ci.V 为当前计数值。

双字对象

执行系统功能时，例如快速计数器 (%FC)、高速计数器 (%HSC) 和脉冲发生器 （%PLS，%PWM），双字对象将提高 Logic Controller 的计算能力。

要对与功能块一起使用的 32 位双字对象进行寻址，只需将字符 “D” 附加到标准字对象的原语法的末尾。

以下示例显示了如何对采用标准格式和双字格式的快速计数器的当前值进行寻址：

- %FCi.V 是采用标准格式的快速计数器的当前值。
- %FCi.VD 是采用双字格式的快速计数器的当前值。

章 3
说明

本章包含了哪些内容？

本章包含了以下部分：

节 3.1
布尔处理

本节目标

本节提供布尔处理指令的简介。

本节包含了哪些内容？

本节包含了以下主题：

布尔指令

简介

布尔指令可比作梯形图语言元素。下表对此类指令进行了概述：

项	操作符	指令示例	描述
测试元素	加载 (LD) 指令相当于连接至梯形图电轨的第一个断开触点。在首个触点连接到梯形图的电源轨之后，逻辑 AND 和 OR 指令等效于断开触点	LD %I0.0	当 %I0.0 位为状态 1 时，触点将闭合。
动作元素	存储 (ST) 指令相当于线圈。	ST %Q0.0	关联位对象使用了位累加器的逻辑值 （上一个逻辑的结果）。

将测试元素的布尔结果应用于动作元素，如以下指令所示：

梯级	指令
0	LD %I0.0 AND %I0.1 ST %Q0.0

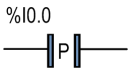
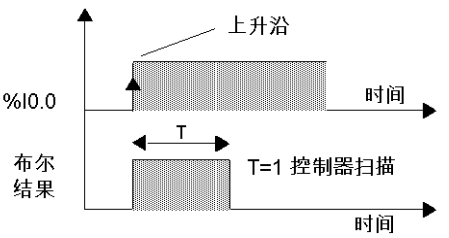
注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

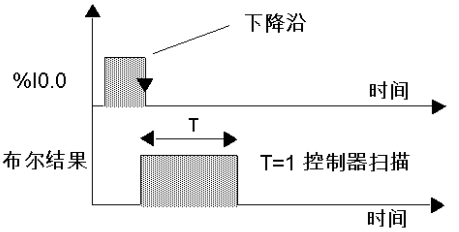
测试控制器输入

可以使用布尔测试指令检测控制器输入上的上升沿或下降沿。输入状态在 “scan n-1” 和当前 “scan n” 之间变化时，检测到沿。该沿在当前扫描期间保持已检测状态。

沿检测

下表对用于检测沿的指令和时序进行了概述：

沿	IL 指令	梯形图	时序图
上升沿	LDR %I0.0		

沿	IL 指令	梯形图	时序图
下降沿	LDF %I0.0	%I0.0 N	

注意： 只可通过 %I 和 %M 应用上升沿和下降沿。

上升沿检测

加载上升沿 (LDR) 指令相当于上升沿检测触点。上升沿检测到从 0 到 1 的输入值变化。

正转换感应触点可用于检测上升沿，如以下示例所示：

梯级	指令
0	LDR %I0.0

注意： 请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

下降沿检测

加载下降沿 (LDF) 指令相当于下降沿检测触点。下降沿检测到从 1 到 0 的控制输入变化。

负转换感应触点可用于检测下降沿，如以下示例所示：

梯级	指令
0	LDF %I0.0

注意： 请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

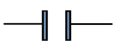
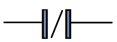
加载操作符 (LD, LDN, LDR, LDF)

简介

加载操作符 LD、 LDN、 LDR 和 LDF 分别对应于断开、关闭、上升沿和下降沿触点。 LDR 和 LDF 仅用于 Logic Controller 输入和内存字。

语法

下表列出了带梯形图等效物的加载操作符和允许的操作数：

操作符	梯形图等效物	许可的操作数
LD		0/1 %I, %Q, %M, %S, %X, %BLK.x %IW:Xk, %QW:Xk, %IWS:Xk, %QWS:Xk, %MW:Xk, %SW:Xk, %KW:Xk
LDN		
LDR		%I 和 %M
LDF		

代码示例

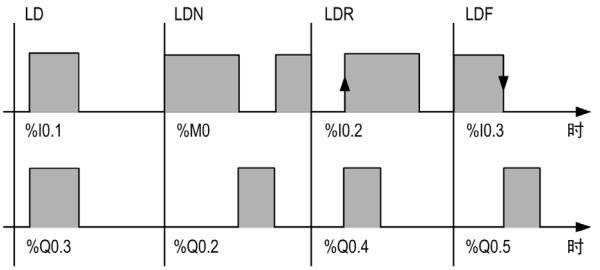
加载指令示例：

梯级	指令
0	LD %I0.1 ST %Q0.3
1	LDN %M0 ST %Q0.2
2	LDR %I0.1 ST %Q0.4
3	LDF %I0.3 ST %Q0.5

注意： 请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

时序图

下图说明了代码示例中的代码的计时以及对输出的影响：



赋值操作符 （ST, STN, R, S)

简介

赋值操作符 ST、 STN、 S 和 R 分别对应于直接线圈、反转线圈、置位线圈和复位线圈。

语法

下表列出了带梯形图等效物的赋值操作符和允许的操作数：

操作符	梯形图等效物	许可的操作数
ST		%Q, %M, %S, %X, %BLK.x %QW:Xk, %MW:Xk, %SW:Xk(1)
STN		
S		
R		
(1) %SW:Xk 位于非只读系统字上。		

代码示例

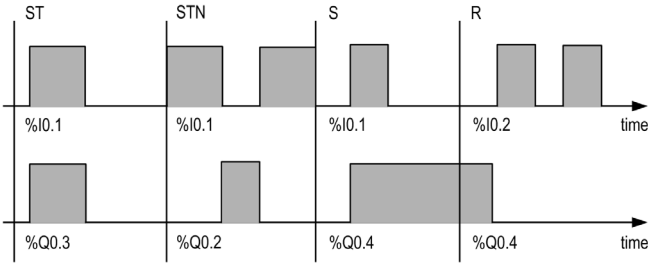
赋值指令示例：

梯级	指令
0	LD %I0.1 ST %Q0.3 STN %Q0.2 S %Q0.4
1	LD %I0.2 R %Q0.4

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

时序图

下图说明了代码示例中的代码的计时以及对输出的影响：



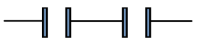
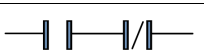
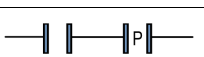
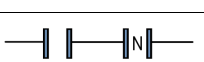
逻辑 AND 操作符 (AND，ANDN，ANDR，ANDF)

简介

AND 操作符在操作数（或其反转、上升沿或下降沿）与前一指令的布尔结果之间执行逻辑 AND 运算。

语法

下表列出了带包含梯形图等效物的 AND 操作符和允许的操作数：

操作符	梯形图等效物	许可的操作数
AND		0/1 %I，%Q，%M，%S，%X，%BLK.x %IW:Xk，%QW:Xk，%IWS:Xk，%QWS:Xk，%MW:Xk，%SW:Xk，%KW:Xk
ANDN		
ANDR		%I，%M
ANDF		

代码示例

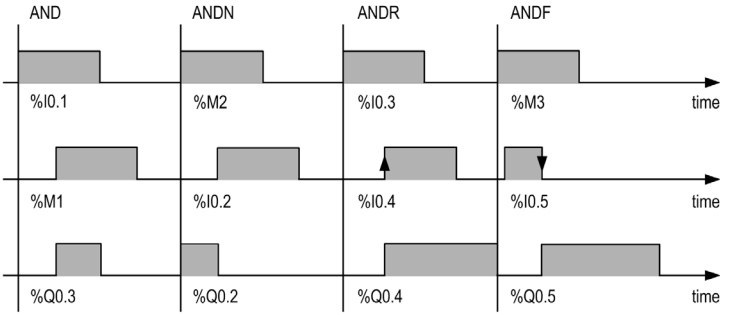
逻辑 AND 指令示例：

梯级	指令
0	LD %I0.1 AND %M1 ST %Q0.3
1	LD %M0 ANDN %I0.0 ST %Q0.2
2	LD %I0.3 ANDR %I0.4 S %Q0.4
3	LD %M3 ANDF %I0.5 S %Q0.5

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

时序图

下图说明了代码示例中的代码的计时以及对输出的影响：



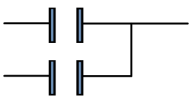
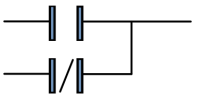
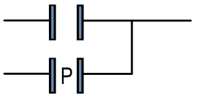
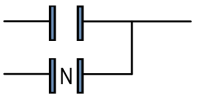
逻辑 OR 操作符 (OR，ORN，ORR，ORF)

简介

OR 操作符在操作数 （或其反转、上升沿或下降沿）与前一指令的布尔结果之间执行逻辑 OR 运算。

语法

下表列出了带梯形图等效物的 OR 操作符和允许的操作数：

操作符	梯形图等效物	许可的操作数
OR		0/1 %I, %Q, %M, %S, %X, %BLK.x %IW:Xk, %QW:Xk, %IWS:Xk, %QWS:Xk, %MW:Xk, %SW:Xk, %KW:Xk
ORN		
ORR		%I, %M
ORF		

代码示例

逻辑 OR 指令示例：

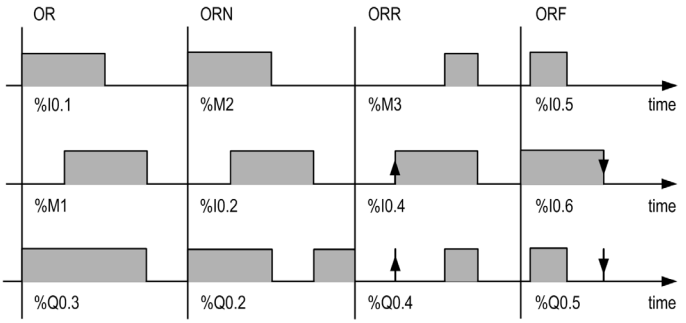
梯级	指令
0	LD %I0.1 OR %M1 ST %Q0.0
1	LD %I0.2 ORN %M2 ST %Q0.1

梯级	指令
2	LD %M0 ORR %I0.3 S %Q0.5
3	LDF %I0.5 ORF %I0.6 S %Q0.0

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

时序图

下图说明了代码示例中的代码的计时以及对输出的影响：



互斥 OR 操作符 (XOR，XORN，XORR，XORF)

简介

XOR 操作符在操作数和操作符指令的布尔结果之间执行异或运算。

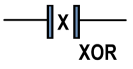
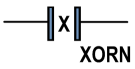
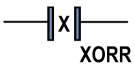
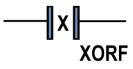
XORN 操作符在操作数反转和前一个指令的布尔结果之间执行异或运算。

XORR 操作符在操作数上升沿和前一个指令的布尔结果之间执行异或运算。

XORF 操作符在操作数下降沿和前一个指令的布尔结果之间执行异或运算。

语法

下表列出了带 XOR 操作符和允许的操作数：

操作符	梯形图等效物	许可的操作数
XOR		%I，%Q，%M，%S，%X，%BLK.x %IW:Xk，%QW:Xk，%IWS:Xk，%QWS:Xk，%MW:Xk，%SW:Xk，%KW:Xk
XORN		
XORR		%I，%M
XORF		

代码示例

使用 XOR 指令：

梯级	指令
0	LD　%I0.1 XOR　%M1 ST　%Q0.3

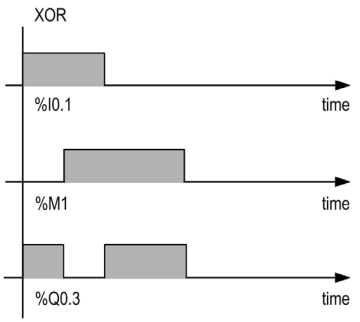
XOR 操作符的等效逻辑指令：

梯级	指令
0	LD %I0.1 ANDN %M1 OR(%M1 ANDN %I0.1) ST %Q0.3

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

时序图

下图说明了代码示例中的代码的计时以及对输出的影响：

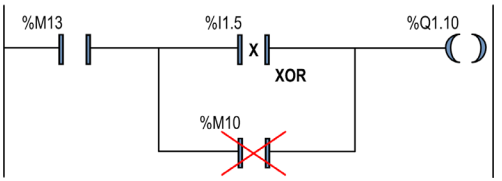


特殊情况

请勿：

- 将 XOR 触点插入到梯级的第一个位置。
- 与其他梯形图元素并行插入 XOR 触点 （请参见以下示例）。

如以下示例所示，将某元素与 XOR 触点并行插入将在验证时生成错误。



NOT 操作符 (N)

简介

NOT (N) 操作符具有隐含操作数：即布尔累加器中保存的结果。 NOT 对累加器值进行否定。

语法

下表显示了 N 操作符：

操作符	梯形图等效物	许可的操作数
N		不适用。

代码示例

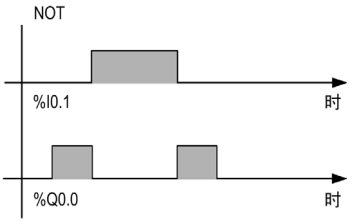
NOT 指令示例：

梯级	指令
0	LD %I0.1 N ST %Q0.0

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

时序图

下图说明了代码示例中的代码的计时以及对输出的影响：



比较指令

简介

比较操作符可用于比较两个操作数。

下表列出了比较操作符的类型：

操作符	功能
>	测试 Op1 是否大于 Op2
>=	测试 Op1 是否大于或等于 Op2
<	测试 Op1 是否小于 Op2
<=	测试 Op1 是否小于或等于 Op2
=	测试 Op1 是否等于 Op2
<>	测试 Op1 是否不等于 Op2

语法

下面介绍了指令列表语法。您可以使用 (参见第 *18* 页) 比较块**图形元素**将指令列表比较表达式插入梯形图梯级。

比较指令的语法：

操作符	语法
>、 >=、 <、 <=、 = 和 <>	LD [Op1 操作符 Op2] AND [Op1 操作符 Op2] OR [Op1 操作符 Op2]

下表列出了操作数的详细信息：

类型	Op1	Op2
字	%MWi, %KWi, %IW, %QWi, %SWi, %BLK.x	立即值, %MWi, %KWi, %IW, %QW, %IWSi, %QWSi, %SWi, %BLK.x, %MWi[%MWi], %KWi[%MWi]
双字	%MDi, %KDi	立即值, %MDi, %KDi, %MDi[%MWi], %KD[%MWi]
浮点字	%MFi, %KFi	立即浮点值, %MFi, %KFi, %MFi[%MWi], %KFi[%MWi]

注意：比较指令可在圆括号内使用。

代码示例

在紧跟指令 LD、 AND 和 OR 后的方括号内执行比较。请求的比较为 True 时，结果为 1。

比较指令示例：

梯级	指令
0	LD %I0.2 AND [%MW10>100] ST %Q0.3
1	LD %M0 AND [%MW20<%KW35] ST %Q0.4
2	LD %I0.2 OR [%MF30>=%MF40] ST %Q0.5

在圆括号内使用比较指令的示例：

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

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

节 3.2
数字处理

本节目标

本节提供数字处理的简介。

本节包含了哪些内容?

本节包含了以下主题：

数字运算简介

概览

数字指令通常适用于 16 位字和 32 位双字。此类指令以方括号括起。若前一逻辑运算的结果为 True （布尔累加器 = 1），则执行数字指令。若前一个逻辑运算的结果为 False （布尔累加器 = 0），则不执行数字指令且操作数保持不变。

赋值指令

简介

赋值指令用于将 Op2 （操作数 2）载入 Op1 （操作数 1）。

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

赋值指令的语法：

操作符	语法
:=	[Op1: = Op2] Op1 将接受 Op2 的值

可对下列对象执行赋值操作：

- 位字符串
- 字
- 双字
- 浮点字
- 字表
- 双字表
- 浮点字表

位字符串赋值

简介

可对下列位字符串进行运算：

- 位字符串到位字符串 （示例 1）
- 位字符串到字 （示例 2）或双字 （带索引）
- 字或双字 （带索引）到位字符串 （示例 3）
- 立即值到位字符串

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

位字符串赋值的语法：

操作符	语法
:=	[Op1: = Op2] Op1 将接受 Op2 的值

下表列出了操作数的详细信息：

类型	Op1	Op2
字、双字	%MWi,%QWi, %SWi %MWi[%MWi], %MDi, %MDi[%MWi] %Mi:L, %Qi:L, %Si:L, %Xi:L %TMi.P, %Ci.P, %Ri.I, %Ri.O, %FCi.P, %PLSi.P, %PWMi.P %Ci.PD, %FCi.PD	立即值， %MWi, %KWi, %IW, %QWi, %IWSi, %QWSi, %SWi,%BLK.x, %MWi[%MWi], %KWi[%MWi], %MDi[%MWi], %KDi[%MWi], %Mi:L,%Qi:L, %Si:L, %Xi:L, %Ii:L %TMi.P, %Ci.P, %Ri.I, %Ri.O, %FCi.P, %PLSi.P, %PWMi.P %Ci.PD, %FCi.PD

注意：缩写 %BLK.x （例如， %C0.P）可用于描述任意功能块字。

结构

位字符串赋值的示例：

梯级	指令
0	LD 1 [%Q0.0:8:=%M64:8]
1	LD %I0.2 [%MW100:=%M0:16]
2	LDR %I0.3 [%MW104:16:=%KW0]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

使用规则：

- 对于位字符串到字的赋值：从右侧开始，将字符串的位传输到字 （字符串的第一位传输到字中的位 0），并将传输中未使用的字位 （长度 ≤16）设置为 0。
- 对于字到位字符串的赋值：从右侧开始传输字位 （字的位 0 传输到字符串中的第一位）。

字赋值

简介

可对以下字和双字执行赋值操作：

- 字 （带索引）到字 （示例 2）（带索引或不带索引）
- 双字 （带索引）到双字 （带索引或不带索引）
- 整数立即值到字 （示例 3）或双字 （带索引或不带索引）
- 位字符串到字或双字
- 浮点数 （带索引或不带索引）到浮点数 （带索引或不带索引）
- 字或双字到位字符串
- 浮点数立即值到浮点数 （带索引或不带索引）

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

字赋值的语法：

操作符	语法
:=	[Op1: = Op2] Op1 将接受 Op2 的值

下表列出了操作数的详细信息：

类型	Op1	Op2
字、双字和位字符串	%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, %IWSi, QWSi, %SWi, %MWi[MWi], %KWi[MWi], %MDi, %MDi[%MWj], %KDi, %KDi[MWj], %Mi:L, %Qi:L, %Si:L, %Xi:L, %Ii:L
浮点数	%MFi, %MFi[%MWj]	立即浮点值， %MFi, %MFi[%MWj], %KFi, %KFi[%MWj]

注意：缩写 %BLK.x （例如， R3.I）可用于描述任意功能块字。对于位字符串 %Mi:L、 %Si:L 和 %Xi:L，位字符串首位的基址必须是 8 的倍数 （0、 8、 16、 ...、 96、 ...）。

结构

字赋值的示例：

梯级	指令
0	LD 1 [%SW112:=%MW100]
1	LD %I0.2 [%MW0[%MW10]:=%KW0[%MW20]]
2	LD %I0.3 [%MW10:=100]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

整数算术操作符

简介

算术操作符可用于在两个整数操作数之间或对一个整数操作数执行算术运算。

下表列出了算术操作符的类型：

操作符	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	两个操作数相除后求余数
SQRT	操作数的平方根
INC	操作数累加
DEC	操作数累减
ABS	操作数的绝对值

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

算术指令的语法：

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

下表列出了操作数的详细信息：

类型	Op1	Op2 和 Op3(1)
字	%MWi, %QWi, %SWi, %BLK.x(2)	立即值, %MWi, %KWi, %IWi, %QWi, %IWSi, %QWSi, %SWi, %BLK.x(2)
双字	%MDi, %BLK.x	Immediate value, %MDi, %KDi, %BLK.x(2)
(1) 使用此操作符时， Op2 不得为立即值。 ABS 函数仅适用于双字 （%MD 和 %KD）和浮点数 （%MF 和 %KF）。因此， Op1 和 Op2 必须为双字或浮点数。 **(2)** %BLK.x 表示所有块对象。		

结构

算术指令示例：

梯级	指令
0	LD %M0 [%MW0:=%MW10+10]
1	LD %I0.2 [%MW0:=SQRT(%MW10)]
2	LDR %I0.3 [%MW10:=32767]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

特殊情况

加法

- 字运算期间溢出
  若结果超出结果字的容量，位 %S18 （溢出位）将设置为 1，且结果无效 （参见应用示例 (参见第 *67* 页) 的梯级 1）。用户程序负责管理位 %S18。

注意：对于双字，最小和最大值分别为 -2147483648 和 2147483647。

乘法

- 运算期间溢出
  若结果超出结果字的容量，位 %S18 （溢出位）将设为 1，且结果无效。

除法 / 求余数

- 除零
  若除数为 0，则无法执行除法，且系统位 %S18 将设为 1，因而结果错误。
- 运算期间溢出
  若除法的商超过结果字的容量，则位 %S18 将设为 1。

开平方

- 运算期间溢出
  只能对正值开平方。因此，结果始终为正数。若平方根操作数为负数，则系统位 %S18 将设为 1，且结果错误。

检测到的一些数学错误可能会对应用程序的执行产生重大影响。您有责任监控这些潜在的错误，如果出现一个或多个上述检测到的错误，则必须对指令进行编程以适当控制应用程序的执行。任何上述检测到的错误的影响取决于配置、所使用的设备以及在检测到潜在错误之前和之后所执行的程序指令。

⚠ 警告

意外的设备操作

- 写入编辑指令，以测试用于数学运算的操作数是否有效。
- 避免在数学运算中使用不同数据类型的操作数。
- 始终监控分配的系统位以指示无效的数学结果。

不遵循上述说明可能导致人员伤亡或设备损坏。

注意：用户程序负责管理系统位 %S17 和 %S18。它们由控制器设为 1，且必须由程序重置，以便重复使用 （参见上一页的示例）。

应用示例

加法期间溢出：

梯级	指令
0	LD %M0 [%MW0:=%MW1+%MW2]
1	LDN %S18 [%MW10:=%MW0]
2	LD %S18 [%MW10 :=32767]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

如果 %MW1 =23241，同时 %MW2=21853，则结果会是 (45094)，不能用 1 个带符号的 16 位字表示。因此，位 %S18 设置为 1，用 %MW0 (-20442) 表示的值不正确。在本例中，当结果大于 32767 时，其值固定为 32767。

逻辑指令

简介

逻辑操作符可用于在 2 个字操作数之间 （或者对于逻辑 NOT ， 在 1 个字操作数上）执行逻辑运算。

下表列出了逻辑指令的类型：

指令	功能
AND	2 个操作数之间的 AND （按位操作）
OR	2 个操作数之间的逻辑 OR （按位操作）
XOR	2 个操作数之间的异或 (OR) （按位操作）
NOT	操作数的逻辑反码 （按位操作）

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

逻辑指令的语法：

操作符	语法	Op1	Op2 和 Op3
AND, OR, XOR	[Op1: = Op2 操作符 Op3]	%MWi, %QWi, %SWi, %BLK.x	立即值 (1), %MWi, %KWi, %IWi, %QWi, %IWSi, %QWSi, %SWi, %BLK.x
NOT	[Op1:=NOT(Op2)]		
(1) 使用 NOT 时， Op2 不能为立即值。			

结构

逻辑指令示例：

梯级	指令
0	LD %M0 [%MW0:=%MW10 AND 16#00FF]
1	LD 1 [%MW0:=%KW5 OR %MW10]
2	LD %I0.3 [%MW102:=NOT(%MW100)]

注意： 请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

应用示例

逻辑 AND 指令：

```
[%MW15:=%MW32 AND %MW12]
```

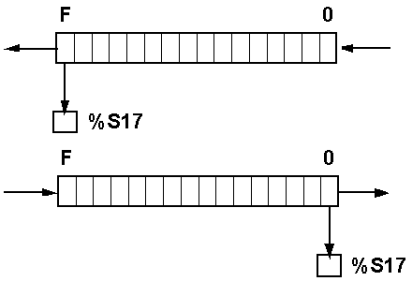
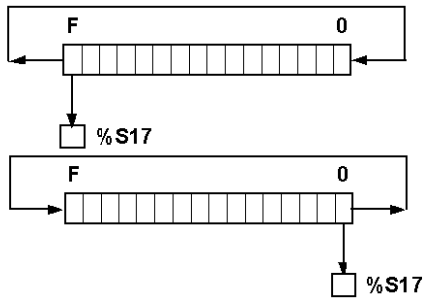
当 %MW32 = 0001 1011 （二进制）（27 （十进制））， 同时 %MW12 = 0011 0110 （二进制）（54 （十进制））时，结果会是 %MW15 = 0001 0010 （二进制）（18 （十进制））。

移位指令

简介

移位指令将操作数的位向右或向左移动指定数量的位置。

下表列出了移位指令的类型：

指令	功能	
逻辑移位		
SHL(op2,n)	向左 n 个位置的逻辑移位。	
SHR(op2,n)	向右 n 个位置的逻辑移位。	
循环移位		
ROL(op2,n)	向左 n 个位置的循环移位。	
ROR(op2,n)	向右 n 个位置的循环移位。	
否 整数立即值： ● 字：1...16（含） ● 双字：1...32（含）。		

注意： 系统位 %S17 表示上一次弹出的位的值。

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

移位指令的语法：

操作符	语法
SHL、 SHR	[Op1: = 操作符 (Op2,n)]
ROL、 ROR	
否 整数立即值： ● 字：1...16 （含） ● 双字：1...32 （含）。	

下表列出了操作数的详细信息：

类型	Op1	Op2
字	%MWi, %QWi, %SWi %BLK.x	%MWi, %KWi, %IWi, %QWi, %IWSi, %QWSi, %SWi, %BLK.x
双字	%MDi %BLK.x	%MDi, %KDi %BLK.x

结构

移位指令示例：

梯级	指令
0	LDR %I0.1 [%MW0:=SHL(%MW10,5)]
1	LDR %I0.2 [%MW10:=ROR(%KW9,8)]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

二进码十进制 / 二进制转换指令

简介

转换指令在数字的不同表示法之间执行转换。

下表列出了二进码十进制 / 二进制转换指令的类型：

指令	功能
BTI	二进码十进制转换为二进制
ITB	二进制转换为二进码十进制

BCD 码概述

二进码十进制 (BCD) 通过对四个二进制位进行编码以表示十进制数（0 到 9）。因此，一个 4 位字对象可以包含一个以四位数字 （0000 到 9999）表示的数，一个 32 位双字对象则可包含八位数字。

转换期间，若该值不为 BCD 码，则系统位 %S18 将设为 1。该位必须由程序检测并重置为 0。

十进制数字的 BCD 表示法：

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 表示 BCD 值 “2450”，其对应的二进制值为：0010 0100 0101 0000
- 字 %MW12 表示十进制值 “2450”，其对应的二进制值为：0000 1001 1001 0010

字 %MW5 通过使用指令 BTI 转换为字 %MW12。

字 %MW12 通过使用指令 ITB 转换为字 %MW5。

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

二进码十进制 / 二进制转换指令的语法：

操作符	语法
BTI、 ITB	[Op1: = 操作符 (Op2)]

下表列出了操作数的详细信息：

类型	Op1	Op2
字	%MWi, %QWi, %SWi %BLK.x	%MWi, %KWi, %IWi, %QWi, %IWSi, %QWSi, %SWi, %BLK.x
双字	%MDi %BLK.x	%MDi, %KDi %BLK.x

结构

二进码十进制 / 二进制转换指令的示例：

梯级	指令
0	LD %M0 [%MW0:=BTI(%MW10)]
1	LD %I0.2 [%MW10:=ITB(%KW9)]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

应用示例

BTI 指令用于通过 BCD 编码拇指轮来处理控制器输入处的设置点值。

ITB 指令用于显示有关 BCD 编码显示的数值 （例如，计算结果和功能块的当前值）。

单 / 双字转换指令

简介

下表介绍用于在单字和双字之间执行转换的指令：

指令	功能
LW	提取到字的双字 LSB。
HW	提取到字的双字 MSB。
CONCATW	将两个字连接为一个双字。
DWORD	将 16 位字转换为 32 位双字。

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

单 / 双字转换指令的语法：

操作符	语法	Op1	Op2	Op3
LW, HW	Op1 = 操作符 (Op2)	%MWi	%MDi, %KDi, %BLK.x	[-]
CONCATW	Op1 = 操作符 (Op2, Op3)	%MDi, %KDi, %BLK.x	%MWi, %KWi, 立即值	%MWi, %KWi, 立即值
DWORD	Op1 = 操作符 (Op2)	%MDi, %KDi, %BLK.x	%MWi, %KWi	[-]

结构

单 / 双字转换指令示例：

梯级	指令
0	LD %M0 [%MW0:=HW(%MD10)]
1	LD %I0.2 [%MD10:=DWORD(%KW9)]
2	LD %I0.3 [%MD11:=CONCATW(%MW10,%MW5)]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

节 3.3
程序

本节目标

本节提供了程序指令的简介。

本节包含了哪些内容？

本节包含了以下主题：

END 指令

简介

END 指令定义执行程序扫描的结束。

END、 ENDC 和 ENDCN

共有三种不同的 END 指令可用：

- END：无条件结束程序
- ENDC：若前一测试指令的布尔结果为 1，则程序结束
- ENDCN：若前一测试指令的布尔结果为 0，则程序结束

缺省情况下 （正常模式），当已激活程序结束时，将更新输出并启动下次扫描。

如果为周期扫描，当周期结束时，将更新输出并启动下次扫描。

示例

无条件 END 指令示例：

梯级	指令
0	LD %M1 ST %Q0.1
1	LD %M2 ST %Q0.2
2	END

有条件 END 指令示例：

梯级	指令
0	LD %I0.0 ST %Q0.0
1	LD %I0.1 ST %Q0.1
2	LD %I0.2 ENDC
3	LD %I0.3 ST %Q0.2
4	END

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

NOP 指令

简介

NOP 指令不执行任何操作。该指令可用于在程序中“保留”行，以便后续可插入指令，而无需修改行编号。

跳转指令

简介

跳转指令可用于立即中断某一程序的执行，并从含标签 %Li （i = 最大模块编号）的程序行的后一行继续执行。

JMP、 JMPC 和 JMPCN

共有三种不同的跳转指令可用：

- JMP：无条件程序跳转
- JMPC：若前一逻辑的布尔结果为 1，则程序跳转
- JMPCN：若前一逻辑的布尔结果为 0，则程序跳转

示例

跳转指令示例：

梯级	指令
0	LD %M15 JMPC %L8
1	LD [%MW24<%MW12] ST %Q0.3 JMPC %L12
2	%L8: LD %M12 AND %M13 ST %M12 JMPC %L12
3	LD %M11 S %Q0.0
4	%L12: LD %I0.0 ST %Q0.4

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

指南

- 不允许在圆括号中使用跳转指令，且不得将跳转指令置于指令 AND、 OR 和结束圆括号指令 “)” 之间。
- 该标签仅可置于 LD、 LDN、 LDR、 LDF 或 BLK 指令之前。
- 标签 %Li 的标签编号在程序中仅能定义一次。
- 程序跳转执行到下行流或上行流的编程行。当跳转处于上行流时，必须注意程序扫描时间。延长扫描时间可触发看门狗定时器。

子程序指令

简介

子程序指令可产生一个程序，该程序可执行子程序，然后在调用子程序的点返回主程序。

过程

子程序在自由 POU 中创建。有关创建自由 POU 和子程序以及定义子程序数的信息，请参阅自由 POU *(参见 SoMachine Basic, Operating Guide)*。另请参阅管理 POU *(参见 SoMachine Basic, Operating Guide)*，以了解有关通过任务和梯级来管理 POU 的详细信息。

调用子程序分为 3 个步骤：

1. 若如果前一布尔指令的结果为 1，则 SRn 指令将调用自由 POU SRn 所引用的子程序。
2. 该子程序由自由 POU SRn 引用，其中 n 为子程序的数量。
3. 子程序指令必须写为独立于主程序的自由 POU。

有关子程序的更多信息，请参阅创建周期性任务 *(参见 SoMachine Basic, Operating Guide)*。

示例

含子程序的指令示例：

梯级	指令
0	LD %M15 AND %M5 ST %Q0.0
1	LD [%MW24>%MW12] SR1
2	LD %I0.4 AND %M13 ST %Q0.1 END

子程序指令 (SR1)：

梯级	指令
0 (SR1)	LD %I0.0 ST %Q0.0

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

指南

- 子程序无法调用其他子程序。试图调用自由 POU 中的子程序将会产生已检测到的编译器错误。
- 不允许在圆括号中使用子程序指令，且不得将子程序指令置于指令 AND、 OR 和右括号指令 “)” 之间。
- 在 IL 中的子程序后紧接着调用赋值指令时务必要谨慎。因为子程序可能会更改布尔累加器的内容。因此返回时，其值可能会与调用前不同。

节 3.4
浮点数

本节目标

本节介绍浮点数的高级指令。

本节包含了哪些内容?

本节包含了以下主题：

浮点数对象的算术指令

简介

这些指令用于在两个浮点操作数之间或对一个浮点操作数执行算术运算：

指令	目的
+	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
LOG	以 10 为底数的对数
LN	自然对数
SQRT	操作数的平方根
ABS	操作数的绝对值
TRUNC	浮点值的整数部分
EXP	自然指数
EXPT	整数的实数幂

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

浮点数算术指令的操作符和语法：

操作符	语法
+, - *, /	Op1:=Op2 操作符 Op3
SQRT, ABS, TRUNC, LOG, EXP, LN	Op1:= 操作符 (Op2)
EXPT	Op1:= 操作符 (Op2,Op3)

浮点数算术指令的操作数：

操作符	Op1	Op2	Op3
+, - *, /	%MFi	%MFi, %KFi, 立即值	%MFi, %KFi, 立即值
SQRT, ABS, LOG, EXP, LN	%MFi	%MFi, %KFi	[-]
TRUNC	%MFi, %MDi	%MFi, %KFi	[-]
EXPT	%MFi	%MFi, %KFi	%MWi, %KWi, 立即值
注意：SoMachine Basic 避免使用将 %MWi 作为 Op1 的函数。			

结构

算术指令示例：

梯级	指令
0	LD %M0 [%MF0:=%MF10+129.7]
1	LD %I0.2 [%MF1:=SQRT(%MF10)]
2	LDR %I0.3 [%MF2:=ABS(%MF20)]
3	LDR %I0.4 [%MF3:=TRUNC(%MF2)]
4	LD %M1 [%MF4:=LOG(%MF10)]
5	LD %I0.5 [%MF5:=LN(%MF20)]
6	LD %I0.0 [%MF6:=EXP(%MF30)]
7	LD %I0.1 [%MF7:=EXPT(%MF40,%MW52)]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

使用规则

- 浮点数和整数值的运算不能直接混合。转换运算 (参见第 *88* 页) 可在这些格式之间进行转换。
- 系统位 %S18 的管理方式与整数运算 (参见第 *88* 页) 相同，字 %SW17 指示所检测到的错误的原因。
- 当函数的操作数为无效数 （例如，负数的对数）时，它将产生一个未定或无限大的结果，并将位 %S18 更改为 1。字 %SW17 指示所检测到的错误的原因。

注意：对于 TRUNC 指令，系统位 %S17 不受影响。

带 %MDi 的 TRUNC 指令的应用示例

下表列出了当 %MDi 用于存储结果时 TRUNC 指令的示例：

示例	结果
TRUNC(3.5)	3
TRUNC(324.18765)	324
TRUNC(927.8904)	927
TRUNC(-7.7)	-7
TRUNC(45.678E+20)	2 147 483 647 （最大有符号双字） (1) %S18 设为 1
TRUNC(-94.56E+13)	- 2 147 483 648 （最小有符号双字） (1) %S18 设为 1
(1) 本示例适用于和 %MDi 一起使用时的 TRUNC 指令。（和 %MFi 一起使用时，TRUNC 指令没有溢出，因此没有最大 / 最小限制。）	

三角指令

简介

用户可使用这些指令执行三角运算：

SIN	以弧度表示的角的正弦，	ASIN	反正弦（结果在 $-\frac{\pi}{2}$ 到 $\frac{\pi}{2}$ 范围内）
COS	以弧度表示的角的余弦，	ACOS	反余弦（结果在 0 到 π 范围内）
TAN	以弧度表示的角的正切，	ATAN	反正切（结果在 $-\frac{\pi}{2}$ 到 $\frac{\pi}{2}$ 范围内）

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

三角运算指令的操作符、操作数和语法

操作符	语法	Op1	Op2
SIN, COS, TAN, ASIN, ACOS, ATAN	Op1:= 操作符 (Op2)	%MFi	%MFi, %KFi

结构

Trigonometric 指令示例：

梯级	指令
0	LD %M0 [%MF0:=SIN(%MF10)]
1	LD %I0.0 [%MF1:=TAN(%MF20)]
2	LD %I0.3 [%MF2:=ATAN(%MF30)]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

使用规则

- 当函数的操作数为无效数时 （例如，某一大于 1 的数的余弦值），它将产生一个未定或无限大的结果，并将位 %S18 更改为 1。字 %SW17 指示所检测到的错误的原因。
- 函数 SIN/COS/TAN 允许 -4096π 和 4096π 之间的角度作为参数，但是，由于在任何运算之前在参数上执行的模数 2π 引起的不精确性，在 -2π 和 $+2\pi$ 周期以外角度的精度逐步降低。

角度转换指令

简介

这些指令用于执行转换运算：

DEG_TO_RAD	将角度转换为弧度，结果为 0 和 2π 之间的角度值
RAD_TO_DEG	转换以弧度表示的角，转换结果为 0 到 360 度之间的角度值

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

转换指令的操作符、操作数和语法

操作符	语法	Op1	Op2
DEG_TO_RAD RAD_TO_DEG	Op1:= 操作符 (Op2)	%MFi	%MFi, %KFi

结构

转换指令示例：

梯级	指令
0	LD %M0 [%MF0:=DEG_TO_RAD(%MF10)]
1	LD %M2 [%MF2:=RAD_TO_DEG(%MF20)]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

使用规则

待转换的角度必须介于 -737280.0 和 +737280.0 之间（对于 DEG_TO_RAD 转换）或介于 -4096π 和 4096π 之间 （对于 RAD_TO_DEG 转换）。

对于超出上述范围的值，显示结果都将 + 1。 #QNAN、 %S18 和 %SW17:X0 位将设为 1。

整数 / 浮点数转换指令

简介

共有四条转换指令可用：

INT_TO_REAL	将整数字转换为浮点数
DINT_TO_REAL	将双字 （整数）转换为浮点数
REAL_TO_INT	将浮点数转换为整数字 （结果为最近的代数值）
REAL_TO_DINT	将浮点数转换为整数字 （结果为最近的代数值）

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见 第 *17* 页) 梯形图梯级。

操作符和语法 （将整数字转换为浮点数）：

操作符	语法
INT_TO_REAL	Op1=INT_TO_REAL(Op2)

操作数 （将整数字转换为浮点数）：

Op1	Op2
%MFi	%MWi,%KWi

示例：整数字转换为浮点数：147 转换为 1.47e+02

操作符和语法 （将整数字双转换为浮点数）：

操作符	语法
DINT_TO_REAL	Op1=DINT_TO_REAL(Op2)

操作数 （将整数字双转换为浮点数）：

Op1	Op2
%MFi	%MDi,%KDi

示例：整数双字转换为浮点数：68905000 转换为 6.8905e+07

操作符和语法 （将浮点数转换为整数字或整数双字）：

操作符	语法
REAL_TO_INT	Op1= 操作符 (Op2)
REAL_TO_DINT	

操作符 （将浮点数转换为整数字或整数双字）：

类型	Op1	Op2
字	%MWi	%MFi, %KFi
双字	%MDi	%MFi, %KFi

示例：
- 浮点数转换为整数字：5978.6 转换为 5979
- 浮点数转换为整数双字：-1235978.6 转换为 -1235979

注意：若在实数到整数 （或实数到整数双字）的转换期间，浮点值超过字 （或双字）的限制，则位 %S18 将设为 1。

结构

整数 / 浮点数转换指令示例：

梯级	指令
0	LD 1 [%MF0:=INT_TO_REAL(%MW10)]
1	LD I0.8 [%MD2:=REAL_TO_DINT(%MF9)]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

舍入精度

标准 IEEE 754 定义了四种浮点运算舍入模式。

以上指令所使用的模式为 “ 四舍五入 ” 模式：

“ 如果最近的可表示值与理论结果相差的大小相等，则给定值的低有效位等于 0。 ”

也就是说，值将进行向上或向下取整，但取整为偶数。

例如：
- 值 10.5 的舍入值为 10。
- 值 11.5 的舍入值为 12。

节 3.5
ASCII

本节目标

本节介绍 ASCII 的高级指令。

本节包含了哪些内容?

本节包含了以下主题：

ROUND 指令

简介

ROUND 指令将以 ASCII 字符串形式存储的浮点数表示形式进行舍入。

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

对于 ROUND 指令，请使用语法 `Op1 := ROUND( Op2,Op3 )`。

例如：

```
[%MW0:7:=ROUND(%MW8,4)]
```

参数

下表列出了 ROUND 函数的参数：

参数	描述
Op1	%MW 用于存储结果
Op2	%MW 含有待舍入的浮点数
Op3	舍入中要求的有效位数 从 1 到 8 的整数

使用规则

ROUND 指令规则如下：

- 操作数始终向下舍入。
- 操作数字符串的结束字符用作结果字符串的结束字符。
- 该结束字符可为不在 [0 - 9] ([16#30 - 16#39]) 范围中的任意 ASCII 字符，但下列字符除外：
  - 点 “.”(16#2E)、
  - 减号 “-”(16#2D)、
  - 加号 “+”(16#2B) ；
  - 指数 “e” 或 “E” （16#65 或 16#45）。
- 结果和操作数不应超过 13 个字节：ASCII 字符串的最大大小为 13 个字节。
- 不允许使用科学计数法。

特殊情况

由软件检查语法。下面的示例将导致语法错误：

错误的语法	正确的语法
%MW10:= ROUND(%MW1,4) 结果中缺少 “:7”	%MW10:7 := ROUND(%MW1,4)
%MW10:13:= ROUND(%MW1,4) %MW10:n，其中 n ≠ 7 不正确	%MW10:7 := ROUND(%MW1,4)

应用示例

下表列出了 ROUND 指令示例：

示例	结果
ROUND("987654321", 5)	“987650000”
ROUND("-11.1", 8)	“-11.1”
ROUND("NAN")	“NAN”

ASCII 到整数转换指令

简介

ASCII 到整数转换指令可将 ASCII 字符串转换为整数值。

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

对于 ASCII 到整数转换指令，采用下列语法：`Op1 := ASCII_TO_INT( Op2 )`

例如：

```
[%MW0:=ASCII_TO_INT(%MW8)]
```

参数

下表介绍了 ASCII 到整数转换函数的参数：

参数	描述
Op1	%MW 用于存储结果
Op2	%MW 或 %KW

使用规则

ASCII 到整数转换指令的规则如下：

- Op2 必须介于 -32768 至 32767 之间。
- 该函数始终首先读取最高有效字节。
- 不在 ['0' - '9'] ([16#30 - 16#39]) 范围内的 ASCII 字符均可用作结束字符，但减号 “-”(16#2D) 作为首个字符时除外。
- 在溢出 （>32767 或 <-32768）情况下，系统位 %S18 （算术溢出或检测到的错误）将设为 1，并返回值 32767 或 -32768。
- 如果操作数的第一个字符是 “separator” 字符，则返回值 0，并且位 %S18 将设为 1。
- 不允许使用科学计数法。

应用示例

假定已在 %MW10 到 %MW13 中存储下列 ASCII 数据：

参数	十六进制值	ASCII 值
%MW10	16#3932	9, 2
%MW11	16#3133	1, 3
%MW12	6#2038	' ', 8
%MW13	16#3820	8, ' '

下表显示了 ASCII 到整数转换的示例：

示例	结果
%MW20 := ASCII_TO_INT(%MW10)	%MW20 = 29318
%MW20 := ASCII_TO_INT(%MW12)	%MW20 = 8
%MW20 := ASCII_TO_INT(%MW13)	%MW20 = 0 且 %S18 设为 1

整数到 ASCII 转换指令

简介

整数到 ASCII 转换指令可将整数转换为 ASCII 字符串值。

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

对于整数到 ASCII 的转换指令，采用下列语法：`Op1 := INT_TO_ASCII( Op2 )`

例如：

```
[%MW0:4:=INT_TO_ASCII(%MW8)]
```

参数

下表描述了整数到 ASCII 转换函数的参数：

参数	描述
Op1	%MW 用于存储结果
Op2	%MW, %KW, %SW, %IW, %QW 或任意字 （不接受立即值）

使用规则

整数到 ASCII 的转换规则如下：

- Op2 必须介于 -32768 至 32767 之间。
- 该函数始终首先写入最高有效字节。
- 结束字符为 “Enter”(ASCII 13)。
- 该函数可自动确定应用 ASCII 值填充多少 %MWs （从 1 到 4）。

语法错误

由软件检查语法。下面的示例将导致语法错误：

错误的语法	正确的语法
%MW10 := INT_TO_ASCII(%MW1) 结果中缺少 “:4”	%MW10:4 := INT_TO_ASCII(%MW1)
%MW10:n := INT_TO_ASCII(%MW1) %MW10:n，其中 $n \neq 4$ 不正确	%MW10:4 := INT_TO_ASCII(%MW1)

应用示例

对于指令 MW10:4 := INT_TO_ASCII(%MW1):

如果 ...	则 ...	
整数值	**十六进制值**	**ASCII 值**
%MW1 = 123	%MW10 = 16#3231	2, 1
	%MW11 = 16#0D33	3
%MW1 = 45	%MW10 = 16#3534	5, 4
	%MW11 = 16#000D	‘enter‘
%MW1 = 7	%MW10 = 16#0D37	‘enter’, 7
%MW1 = -12369	%MW10 = 16#3145	1, ‘-’
	%MW11 = 16#3332	3, 2
	%MW10 = 16#3936	9, 6
	%MW11 = 16#000D	‘enter‘

ASCII 到浮点数转换指令

简介

ASCII 到浮点数转换指令可将 ASCII 字符串转换为浮点值。

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

对于 ASCII 到浮点数转换指令，采用下列语法：`Op1 := ASCII_TO_FLOAT( Op2 )`。

例如：

```
[%MF0:=ASCII_TO_FLOAT(%MW8)]
```

参数

下表描述了 ASCII 到浮点数转换函数的参数：

参数	描述
Op1	%MF
Op2	%MW or %KW

使用规则

ASCII 到浮点数的转换规则如下：

- 该函数始终首先读取最高有效字节。
- 不在 ['0' - '9'] ([16#30 - 16#39]) 范围内的任意 ASCII 字符均可用作 “ 结束 ” 字符，但下列字符除外：
  - 点 “.”(16#2E) ；
  - 减号 “-”(16#2D) ；
  - 加号 “+”(16#2B) ；
  - 指数 “e” 或 “E” （16#65 或 16#45）。
- ASCII 字符格式可采用科学计数法 （例如， “-2.34567e+13”）或十进制表示法 （例如， “9826.3457”）
- 在溢出情况下 （计算结果 >3.402824E+38 或 -3.402824E+38）：
  - 系统位 %S18 （算术溢出或检测到的错误）将设为 1，
  - %SW17:X3 设为 1，
  - 并返回值 +/- 1.#INF （+ 或 - 无穷值）。
- 如果计算结果介于 -1.175494E-38 和 1.175494E-38 之间，则结果舍入为 0.0。
- 如果操作数不是一个数字：
  - 值 1。返回 #QNAN，
  - 位 %SW17:X0 设为 1。

应用示例

假定已在 %MW10 到 %MW14 中存储下列 ASCII 数据：

参数	十六进制值	ASCII 值
%MW10	16#382D	8, '-'
%MW11	16#322E	2, '.'
%MW12	16#3536	5, 6
%MW13	16#2B65	'+', 'e'
%MW14	16#2032	' ',2

下表显示了 ASCII 到浮点数转换的示例：

示例	结果
%MF20 := ASCII_TO_FLOAT(%MW10)	%MF20 = -826.5
%MF20 := ASCII_TO_FLOAT(%MW11)	%MF20 = 1.#QNAN
%MF20 := ASCII_TO_FLOAT(%MW12)	%MF20 = 6500.0
%MF20 := ASCII_TO_FLOAT(%MW13)	%MF20 = 1.#QNAN
%MF20 := ASCII_TO_FLOAT(%MW14)	%MF20 = 2.0

浮点数到 ASCII 转换指令

简介

浮点数到 ASCII 转换指令可将浮点值转换为 ASCII 字符串值。

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

对于浮点数到 ASCII 转换指令，采用下列语法：`Op1 := FLOAT_TO_ASCII(Op2)`。

例如：

```
[%MW0:7:=FLOAT_TO_ASCII(%MF8)]
```

参数

下表描述了浮点数到 ASCII 转换函数的参数：

参数	描述
Op1	%MW
Op2	%MF or %KF

使用规则

浮点数到 ASCII 的转换规则如下：

- 该函数始终首先写入最高有效字节；
- 采用转换科学计数法进行表示；
- “ 无穷 ” 或 “ 非数字 ” 结果返回字符串 “NAN” ；
- 结束字符为 “Enter”(ASCII 13) ；
- 该功能自动确定应该用 ASCII 值填充多少 %MW ；
- 转换精度为 6 位
- 不允许使用科学计数法。

语法错误

由软件检查语法。下面的示例将导致语法错误：

错误的语法	正确的语法
`%MW10 := FLOAT_TO_ASCII(%MF1)` 结果中缺少 “:7”	`%MW10:7 := FLOAT_TO_ASCII(%MF1)`
`%MW10:n := FLOAT_TO_ASCII(%MF1)` `%MW10:n`，其中 $n \neq 7$ 不正确	`%MW10:7 := FLOAT_TO_ASCII(%MF1)`

应用示例

对于指令 `%MW10:7 := FLOAT_TO_ASCII(%MF1):`

要转换的数字	结果
1234567800	1.23456e+09
0.000000921	9.21e-07
9.87654321	9.87654
1234	1.234e+03

节 3.6
堆栈操作符

堆栈指令 （MPS、 MRD、 MPP）

简介

堆栈指令可处理线圈的路由。 MPS、 MRD 和 MPP 指令采用名为堆栈 （可存储多达 32 个布尔表达式）的临时存储区域。

注意：这些指令不能在圆括号之间的表达式中使用。

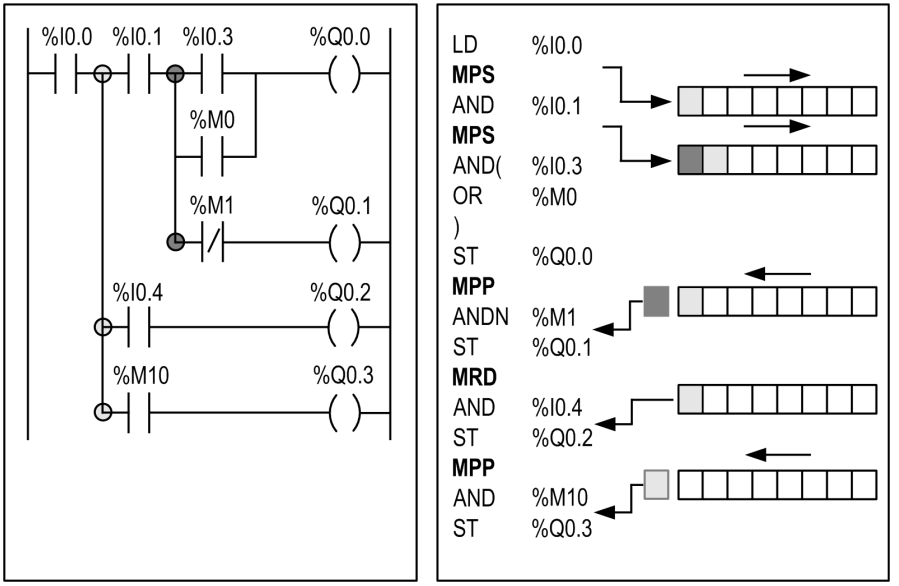
语法

下表介绍了三个堆栈指令：

指令	描述	功能
MPS	内存进栈	将上一个逻辑指令的结果 （累加器的内容）存储到栈顶 （进栈），并将其他值移位到栈底。
MRD	内存读取栈	将栈顶读入累加器。
MPP	内存出栈	将栈顶的值复制到累加器中 （出栈），并将其他值移位到栈顶。

操作

下图显示堆栈指令的运算方式：



应用示例

堆栈指令的使用示例：

梯级	指令
0	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

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

节 3.7
对象表的指令

本节目标

本节介绍下列对象表的指令：

- 双字；
- 浮点对象。

本节包含了哪些内容？

本节包含了以下主题：

字、双字和浮点数表赋值

简介

可对下列对象表执行赋值运算：

- 立即的全值到字表 （参见结构示例 (参见第 *106* 页) 梯级 0）或双字表
- 字到字表 （参见结构示例 (参见第 *106* 页) 梯级 1）
- 字表到字表 （参见结构示例 (参见第 *106* 页) 梯级 2）
  两个表的表长度 (L) 应该相同。
- 双字到双字表
- 双字表到双字表
  两个表的表长度 (L) 应该相同。
- 浮点数立即值到浮点数表
- 浮点数到浮点数表
- 浮点数表到浮点数表
  两个表的表长度 (L) 应该相同。

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

字、双字和浮点数表赋值的语法：

操作符	语法
:=	[Op1: = Op2] Op1 将接受 Op2 的值

下表列出了操作数的详细信息：

类型	Op1	Op2
字表	%MWi:L, %SWi:L	%MWi:L, %SWi:L, 整数立即值 , %MWi, %KWi, %IW, %QW, %SWi, %BLK.x
双字表	%MDi:L	整数立即值 , %MDi, %KDi, %MDi:L, %KDi:L
浮点字表	%MFi:L	立即浮点值 , %MFi, %KFi, %MFi:L, %KFi:L
L 数据表长度 （最多 255 个字符）。		

注意：缩写 %BLK.x （例如， R3.I）可用于描述任意功能块字。

结构

字表赋值示例：

梯级	指令
0	LD 1 [%MW0:10:=100]
1	LD %I0.0 [%MW0:10:=%MW11]
2	LDR %I0.3 [%MW10:20:=%KW20:30]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

表求和函数

简介

SUM_ARR 函数将对象表中的所有元素加在一起：

- 如果表由双字组成，则结果将以双字形式给出；
- 如果表由浮点字组成，则结果将以浮点字形式给出。

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

表求和指令的语法：

```
Res:=SUM_ARR(Tab)
```

表求和指令的参数：

类型	结果 (Res)	表 (Tab)
双字表	%MDi	%MDi:L,%KDi:L
浮点字表	%MFi	%MFi:L,%KFi:L
L 数据表长度 （最多 255 个字符）。		

注意：如果结果不在有效的双字格式范围 （取决于表操作数）之内，则系统位 %S18 将设置为 1。

结构

求和函数示例：

梯级	指令
0	LD %I0.2 [%MD5:=SUM_ARR(%MD3:1)]
1	LD 1 [%MD5:=SUM_ARR(%KD5:2)]
2	LD 1 [%MF2:=SUM_ARR(%MF8:5)]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

应用示例

%MD4:=SUM(%MD30:4)

其中 %MD30=10、 %MD32=20、 %MD34=30、 %MD36=40

因此， %MD4:=10+20+30+40

表比较函数

简介

EQUAL_ARR 函数逐个元素地执行 2 个数据表的比较。

如果显示差异，将以字的形式返回第一对不同元素的序号，否则返回值等于 -1。

比较对整个表执行。

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

表比较指令的语法：

```
Res:=EQUAL_ARR(Tab1,Tab2)
```

表比较指令的参数：

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
L 数据表长度 （最多 255 个字符）。		

注意：数据表必须具有相同的长度和类型。

结构

表比较函数示例：

梯级	指令
0	LD %I0.2 [%MW5:=EQUAL_ARR(%MD20:7,%KD0:7)]
1	LD 1 [%MW0:=EQUAL_ARR(%MD20:7,%KD0:7)]
2	LD 1 [%MF2:=SUM_ARR(%MF8:5)]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

应用示例

```
%MW5:=EQUAL_ARR(%MD30:4,%KD0:4)
```

比较两个表：

序号	字表	常量字表	差别
0	%MD30=10	%KD0=10	=
1	%MD32=20	%KD2=20	=
2	%MD34=30	%KD4=60	不同
3	%MD36=40	%KD6=40	=

字 %MW5 的值为 2（第一对不同元素的序号）

表搜索函数

简介

共有三个搜索函数可用：

- FIND_EQR：在双字表或浮点字表中搜索等于给定值的第一个元素的位置
- FIND_GTR：在双字表或浮点字表中搜索大于给定值的第一个元素的位置
- FIND_LTR：在双字表或浮点字表中搜索小于给定值的第一个元素的位置

这些指令的结果等于找到的第一个元素的序号，如果搜索不成功，则为 -1。

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

表搜索指令的语法：

函数	语法
FIND_EQR	Res:= 函数 (Tab,Val)
FIND_GTR	
FIND_LTR	

浮点字表和双字表搜索指令的参数：

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
L 数据表长度 （最多 255 个字符）。			

结构

表搜索函数示例：

梯级	指令
0	LD %I0.2 [%MW5:=FIND_EQR(%MD20:7,%KD0)]
1	LD %I0.3 [%MW6:=FIND_GTR(%MD20:7,%KD0)]
2	LD 1 [%MW7:=FIND_LTR(%MF40:5,%KF4)]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

应用示例

```
%MW5:=FIND_EQR(%MD30:4,%KD0)
```

在数据表中搜索第一个双字 = %KD0=30 的位置：

序号	字表	结果
0	%MD30=10	-
1	%MD32=20	-
2	%MD34=30	值 (Val)，序号
3	%MD36=40	-

最大值和最小值的表搜索函数

简介

共有两个搜索函数可用：

- MAX_ARR：在双字表和浮点字表中搜索最大值
- MIN_ARR：在双字表和浮点字表中搜索最小值

这些指令的结果等于数据表中找到的最大值 （或最小值）。

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

最大值和最小值的数据表搜索指令语法：

函数	语法
MAX_ARR	结果 := 函数 （选项卡）
MIN_ARR	

最大值和最小值的数据表搜索指令参数：

类型	结果 (Res)	表 (Tab)
双字表	%MDi	%MDn:L,%KDn:L
浮点字表	%MFi	%MFn:L,%KFn:L
i 存储器变量的对象实例标识符。 **否** 表示搜索基本地址的数据表的内存索引。 **L** 搜索方面所要考虑的位置数，包括基本地址索引 （L 的最大值是 255 个字符。		

注意： L 仅对搜索过程中不重叠的地址进行计数。有关详细信息，请参阅对象之间可能重叠的情况 (参见第 *31* 页)。

结构

表搜索函数示例：

梯级	指令
0	LD %I0.2 [%MD0:=MIN_ARR(%MD20:7)]
1	LD 1 [%MF8:=MIN_ARR(%MF40:5)]

注意： 请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

某个值在表中的出现次数

简介

此函数 OCCUR_ARR 在双字表或浮点字表中搜索等于给定值的元素个数。

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

最大值和最小值的表搜索指令语法：

函数	语法
OCCUR_ARR	Res:= 函数 (Tab,Val)

最大值和最小值的表搜索指令参数：

类型	结果 (Res)	表 (Tab)	值 (Val)
双字表	%MWi	%MDi:L,%KDi:L	%MDi,%KDi
浮点字表	%MFi	%MFi:L,%KFi:L	%MFi,%KFi
L 数据表长度 （最多 255 个字符）。			

结构

出现次数的示例：

梯级	指令
0	LD %I0.3 [%MW5:=OCCUR_ARR(%MF20:7,%KF0)]
1	LD %I0.2 [%MW5:=OCCUR_ARR(%MD20:7,%MD1)]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

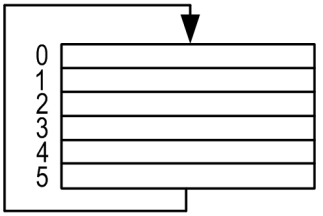
表循环移位函数

简介

共有两个移位函数可用：

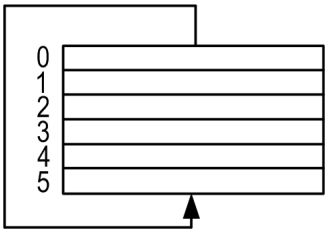
- ROL_ARR：在浮点字表中执行元素的从上到下 n 个位置的循环移位

ROL_ARR 函数示意图



- ROR_ARR：在浮点字表中执行元素的从下到上 n 个位置的循环移位

ROR_ARR 函数示意图



语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

浮点字表或双字表中循环移位指令的语法：ROL_ARR 和 ROR_ARR

函数	语法
ROL_ARR	函数 (n,Tab)
ROR_ARR	

浮点字表的循环移位指令的参数：**ROL_ARR** 和 **ROR_ARR**：

类型	位置数 (n)	表 (Tab)
浮点字表	%MWi, immediate value	%MFi:L
双字表	%MWi, immediate value	%MDi:L
L 数据表长度（最多 255 个字符）。		

注意：如果 n 的值为负或空，则不执行移位。

结构

表循环移位函数示例：

梯级	指令
0	LD %I0.2 [ROL_ARR(%KW0,%MD20:7)]
1	LD %I0.3 [ROR_ARR(2,%MD20:7)]
2	LD %I0.4 [ROR_ARR(2,%MF40:5)]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

表排序函数

简介

排序函数 SORT_ARR 对双字表或浮点字表的元素执行升序或降序排序，并将结果存储在同一表中。

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

表排序函数的语法：

函数	语法
SORT_ARR	函数 (方向 ,Tab)

“ 方向 ” 参数可指定排序的顺序：

- 方向 > 0：以升序进行排序。
- 方向 < 0：以降序进行排序。
- 方向 = 0：未执行排序

结果 （已排序的表）将返回到 Tab 参数 （待排序的表）。

表排序函数的参数：

类型	排序方向	表 (Tab)
双字表	%MWi, immediate value	%MDi:L
浮点字表	%MWi, immediate value	%MFi:L
L 数据表长度 （最多 255 个字符）。		

结构

表排序函数示例：

梯级	指令
0	LD %I0.1 [SORT_ARR(%MW20,%MF0:6)]
1	LD %I0.2 [SORT_ARR(%MW20,%MF0:6)]
2	LD %I0.3 [SORT_ARR(0,%MF40:8)]

注意： 请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

浮点数表插值 (LKUP) 函数

简介

LKUP 函数用于为给定的 X 值内插一组 X 对 Y 浮点数数据。

检查线性插值

LKUP 函数使用线性插值规则，如下列等式的定义：

$$Y = Y_i + \left[\frac{(Y_{i+1} - Y_i)}{(X_{i+1} - X_i)} \cdot (X - X_i)\right] \quad \text{（等式 1）}$$

对于 $X_i \le X \le X_{i+1}$ ，其中 $i = 1 \ldots (m-1)$ ；

假设 X_i 值以升序排列：$X_1 \le X_2 \le \ldots X \ldots \le X_{m-1} \le X_m$ 。

注意：若任意两个连续的 Xi 值相等 （$X_i=X_{i+1}=X$），则等式 (1) 将产生无效例外。在此情况下，要克服该例外，请使用下面的算法代替等式 (1)：

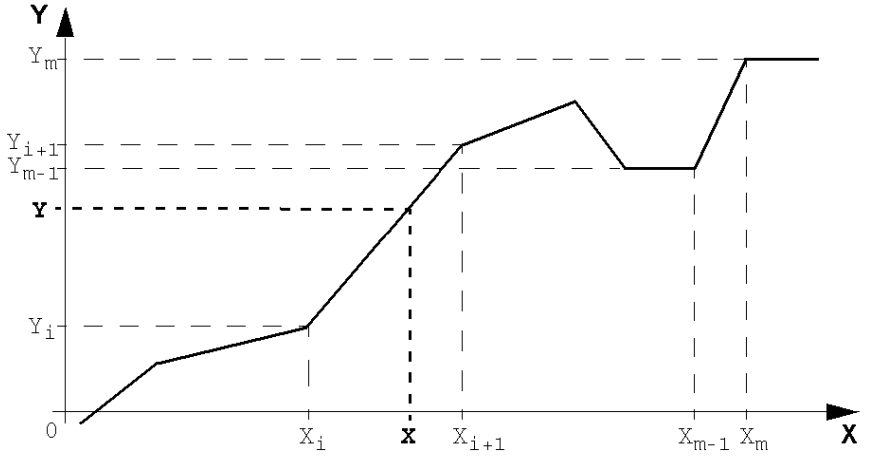
注意：

$$Y = \left[\frac{(Y_{i+1} - Y_i)}{2}\right] \quad \text{（等式 2）}$$

注意：对于 $X_i = X_{i+1} = X$ ，其中 $i = 1 \ldots (m-1)$ 。

图形表示形式

下图对上述线性插值规则进行了说明：



语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

LKUP 函数使用三个操作数，其中两个为函数属性，如下表所述：

语法	**Op1** **输出变量**	**Op2** **用户定义的 (X) 值**	**Op3** **用户定义的 (X_i,Y_i) 变量数组**
[Op1: = LKUP(Op2,Op3)]	%MWi	%MF0	整数值，%MWi，or %KWi

Op1 的定义

Op1 是包含内插函数输出变量的存储器字。

根据 Op1 的值，用户可了解插值是否成功；如果失败，还可了解失败原因，如下表所述：

Op1 (%MWi)	描述
0	插值成功
1	插值检测到的错误：错误数组， $X_m < X_{m-1}$
2	插值检测到的错误：Op2 超出范围， $X < X_1$
4	插值检测到的错误：Op2 超出范围， $X > X_m$
8	数据数组大小无效： ● Op3 被设为奇数，或 ● Op3 < 6。

注意：Op1 不包含计算的插值 (Y)。对于给定的 (X) 值，插值的结果包含在 Op3 数组 (参见第 *119* 页) 的 %MF2 中。

Op2 的定义

Op2 是包含用户定义的用于计算内插 (Y) 值的 (X) 值的浮点数变量 （Op3 浮点数数组的 %MF0）。

Op2 的有效范围： $X_1 \le Op2 \le X_m$ 。

Op3 的定义

Op3 设置存储 (X_i,Y_i) 数据对的浮点数数组的大小 (Op3/2)。

X_i 和 Y_i 数据存储在具有偶索引的浮点数对象中，从 %MF4 处开始 （请注意， %MF0 和 %MF2 浮点数对象分别专用于用户设定点 X 和内插值 Y）。

若为 (m) 数组对的数组 (X_i,Y_i)，使用下列关系设置浮点数数组 (%MFu) 的上索引：

- $Op3 = 2 \cdot m$ （等式 *3*）
- $u = 2 \cdot (Op3 - 1)$ （等式 *4*）

浮点数数组 Op3 (%MFi) 具有与下列示例 （其中 Op3=8）的结构类似的结构：

(X)		(X_1)		(X_2)		(X_3)	
%MF0		%MF4		%MF8		%MF12	
	%MF2		%MF6		%MF10		%MF14
	(Y)		(Y_1)		(Y_2)		(Y_3)
							(Op3=8)

注意：由于以上浮点数数组结构， Op3 必须同时满足下列两个要求，否则将触发 LKUP 函数的检测错误：

- Op3 为偶数，且
- Op3 ≥ 6 （因为至少要有两个数据点才能进行线性插值）。

结构

插值操作执行如下：

梯级	指令
0	LD %I0.2 [%MW20:=LKUP(%MF0,%KW1)]
1	LD %I0.3 [%MW22:=LKUP(%MF0,10)]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

应用示例

使用 LKUP 插值函数：

```
[%MW20:=LKUP(%MF0,10)]
```

在本例中：

- %MW20 为 Op1 （输出变量）。
- %MF0 为用户定义的 (X) 值，其对应的 (Y) 值必须通过线性插值计算。
- %MF2 存储由线性插值产生的计算值 (Y)。
- 10 为 Op3（由上面的 *等式 3* 给定）。它设置浮点数数组的大小。最高序号项 %MFu，其中 u=18 由上面的 *等式 4* 给定。

Op3 数组 [%MF4, . . .%MF18] 中存有四对数据点：

- %MF4 包含 X_1， %MF6 包含 Y_1。
- %MF8 包含 X_2， %MF10 包含 Y_2。
- %MF12 包含 X_3， %MF14 包含 Y_3。
- %MF16 包含 X_4， %MF18 包含 Y_4。

浮点数表的值的 MEAN 函数

简介

MEAN 函数用于计算浮点数表中给定数量的值的平均值。

语法

下面介绍了指令列表语法。您可以使用**操作块**图形元素将指令列表操作和分配指令插入 (参见第 *17* 页) 梯形图梯级。

浮点数表平均值计算函数的语法：

函数	语法
MEAN	结果 = 函数 (Op1)

用于来自浮点数表的给定 L 个值 （最多 255）的计算函数的参数：

Op1	结果 (Res)
%MFi:L, %KFi:L	%MFi

结构

求平均值函数的示例：

梯级	指令
0	LD %I3.2 [%MF0:=MEAN(%MF10:5)]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

章 4
软件对象

本章包含了哪些内容？

本章包含了以下部分：

节 4.1
使用功能块

本节包含了哪些内容?

本节包含了以下主题:

功能块编程原理

概述

功能块是一种可接受一个或多个输入值并返回一个或多个输出值的可重复使用对象。

出现下列情况时，功能块参数无效：

- 控制器不支持功能块；
- 功能块尚未配置。

梯形图程序

在梯形图程序中使用功能块：

1. 将功能块插入 (参见第 *127* 页) 到梯级；
2. 按需连接输入和输出；
3. 通过为功能块的参数赋值以配置 (参见第 *129* 页) 功能块。

指令列表程序

要将功能块添加到指令列表程序，请采用下列任一方法：

- 功能块指令 （例如， BLK %TM2）：该可逆编程法允许在程序内的单一位置对块进行操作。
- 具体指令 （例如， CU %Ci）。非可逆法则允许在程序内多个位置对功能块输入进行操作。例如：

行	指令
1000	CU %C1
1074	CD %C1
1209	R %C1

使用指令 BLK、 OUT_BLK 和 END_BLK 实现功能块的可逆编程：

- BLK：表示块的开始。
- OUT_BLK：用于直接连接块输出。
- END_BLK：表示块的结束。

注意：相关块上的测试和输入指令只能置于 BLK 和 OUT_BLK 指令之间 （或未编程 OUT_BLK 时，置于 BLK 和 END_BLK 之间）。

带输出接线的示例

本示例介绍带已连接输出的程序中的 Counter 功能块：

梯级	指令
0	BLK %C8 LDF %I0.1 R LD %I0.1 AND %M0 CU OUT_BLK LD D AND %M1 ST %Q0.0 END_BLK

注意： 请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

不带输出接线的示例

本示例显示未连接输出的 Counter 功能块的可逆编程：

梯级	指令
0	BLK %C8 LDF %I0.1 R LD %I0.2 AND %M0 CU END_BLK
1	LD %C8.D AND %M1 ST %Q0.4

注意： 请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

添加功能块

向梯形图程序中插入功能块

请完成下列步骤：

步骤	操作
1	在 SoMachine Basic 的编程工作空间内创建新梯形图梯级。有关详细信息，请参阅 SoMachine Basic 操作指南 *(参见 SoMachine Basic, Operating Guide)*。
2	单击位于编程工作空间顶部图形工具栏上的**功能**按钮。 **结果**：显示所有可用软件对象的列表 （参见下表）。
3	选择功能块。
4	将功能块移至梯级中的所需位置，然后单击以将其插入。

可用功能块对象

可用软件对象如下表所示：

软件对象	描述
	Timer
	LIFO/FIFO Register
	Shift Bit Register
	Step Counter
123	Counter
1123	Fast Counter
11123	High Speed Counter
	Drum Register

软件对象	描述
	Pulse
	Pulse Width Modulation
	Message

匹配功能块

在梯形图程序中配置功能块

请完成下列步骤：

步骤	操作
1	单击功能块中的 **[地址]** 标签。 缺省地址随即出现在文本框中，例如 Timer 功能块的 “%TM0”。 要更改缺省地址，请删除地址的最后一个数字 （实例标识符）。 随即出现可用地址的列表。 选择用于标识该功能块实例的地址。 功能块的属性随即出现在功能块对象的中央以及位于编程工作空间下半部的**属性**表中。 在其他任意时间，在功能块内双击任意位置即可显示属性。
2	此外，还可以单击功能块中的 **[输入注释]** 标签，然后键入功能块的简短说明。例如，**脉冲定时器**。
3	此外，还可以单击功能块中的 **[符号]** 标签，然后开始键入符号的名称以与该功能块进行关联。 现有符号中凡以您所键入的一个或多个字符打头的名称的列表随即出现。单击所需符号以便使用。 要为此功能块新建符号，请建议待创建符号的名称，然后选择对象以与该符号进行关联。 有关符号使用的详细信息，请参阅 SoMachine Basic 操作指南 *(参见 SoMachine Basic, Operating Guide)*。
4	按单个功能块描述中 “ 参数 ” 部分的描述，对每个功能块的可用参数进行配置。

注意：通过在梯级中双击功能块，还可显示**属性**表。

节 4.2
定时器 (%TM)

使用定时器功能块

本节介绍如何使用 Timer 功能块并提供其编程指南。

本节包含了哪些内容？

本节包含了以下主题：

描述

简介

Timer 功能块 用于指定执行某操作 （例如，触发某一事件）前的一段时间。

示意图

本示意图为 Timer 功能块。

输入

Timer 功能块具有以下输入：

标签	描述	值
IN	输入地址 （或指令）	检测到上升沿 （TON 或 TP 类型）或下降沿 （TOF 类型）时启动 Timer。

输出

Timer 功能块具有以下输出：

标签	描述	值
Q	输出地址 (%TMi.Q)	当 Timer 过期时，关联位 %TMi.Q 设置为 1（取决于 Timer 类型）。

配置

参数

要配置参数，请执行配置功能块过程 (参见第 *129* 页)，并阅读 SoMachine Basic 操作指南中存储器分配模式的描述。

Timer 功能块具有以下参数：

参数	描述	值
已使用	已使用的地址	如果选择此参数，则当前在程序中使用此地址。
地址	Timer 对象地址 (%TMi)	程序只能包含数量有限的 Timer 对象。有关定时器的最大数量，请参阅相关平台的编程指南。
符号	符号	与此对象关联的符号。有关详细信息，请参阅 SoMachine Basic 操作指南，定义和使用符号。
类型	Timer 类型	下列情况之一： ● TON (参见第 *133* 页)：接通延迟定时器 （缺省值） ● TOF (参见第 *134* 页)：断开延迟定时器 ● TP (参见第 *135* 页)：脉冲定时器 （单稳态）
基数	时基	定时器的时基单位。定时器的基数单位越小，定时器的精度越高。 ● **1 毫秒** （仅适用于前面 6 个实例） ● **10 毫秒** ● **100 毫秒** ● **1 秒** ● **1 分钟**； （缺省值）
预设	预设值	0 至 9999。缺省值为 9999。 定时器周期 = 预设 x 时基 定时器延迟 = 预设 x 时基 使用相关的对象 %TMi.P 可对已配置的此预设值进行读取、测试和修改。
注释	注释	可以将注释与此对象进行关联。

对象

Timer 功能块具有以下对象：

对象	描述	值
%TMi.P	预设值	请参见上表中的 “ 参数 ” 说明。
%TMi.V	当前值	当定时器运行时，字的值从 0 递增至预设值 %TMi.P。程序可以读取和测试该字的值，但不能写入。 可以在动态数据表中修改该字的值。
%TMi.Q	Timer 输出	请参见上表中的 “ 输出 ” 说明。

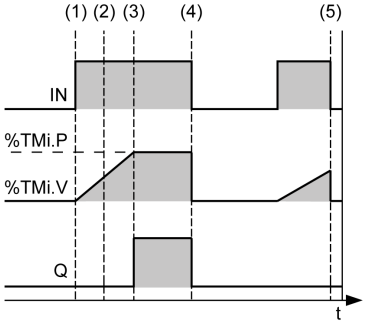
TON：接通延迟定时器

简介

TON (On-Delay Timer) 类型定时器用于控制接通延迟动作。用户可借助软件对该延迟进行编程。

时序图

下图说明 TON 类型 Timer 的操作。



(1) Timer 在 IN 输入的上升沿上启动。
(2) 当前值 %TMi.V 将以时基参数 TB 的每个脉冲的 1 个单位递增，从 0 增加到 %TMi.P
(3) 在当前值达到预设值 %TMi.P 时， %TMi.Q 输出位将设为 1
(4) 当 IN 输入处于 1 时， %TMi.Q 输出位保持在 1 处
(5) 在 IN 输入、 Timer 已停止，即便此时 Timer 尚未达到 %TMi.P。 %TMi.V 将设为 0

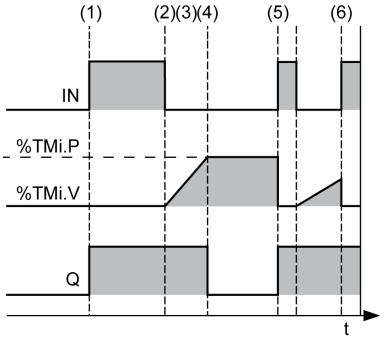
TOF：断开延迟定时器

简介

使用 TOF (Off-Delay Timer) 类型 Timer 可控制断开延迟动作。用户可借助软件对该延迟进行编程。

时序图

下图说明 TOF 类型 Timer 的操作。



(1) 在 IN 输入、 %TMi.Q 将设为 1
(2) Timer 在输入 IN 的下降沿上启动。
(3) 当前值 %TMi.V 将以时基参数 TB 的每个脉冲的 1 个单位递增，增加到预设值 %TMi.P
(4) 在当前值达到预设值 %TMi.P 时， %TMi.Q 输出位将重置为 0
(5) 在输入 IN 的上升沿，将 %TMi.V 设置为 0
(6) 在输入 IN 的上升沿，将 %TMi.V 设置为 0，即便此时尚未达到预设值

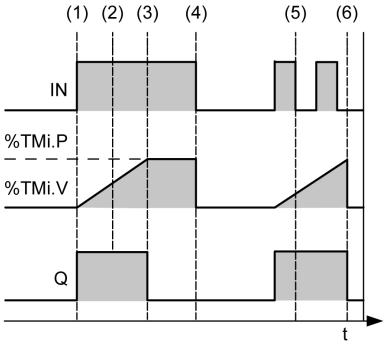
TP：脉冲定时器

简介

TP (Pulse Timer) 类型 Timer 用于创建精确时段的脉冲。用户可借助软件对该延迟进行编程。

时序图

下图说明 TP 类型 Timer 的操作。



(1) Timer 在 IN 输入的上升沿上启动。如果 Timer 尚未启动，当前值 %TMi.V 将设为 0， Timer 启动时，%TMi.Q 将设为 1
(2) Timer 的当前值 %TMi.V 以时基参数 TB 每个脉冲的 1 个单位递增，从 0 增加到预设值 %TMi.P
(3) 在当前值达到预设值 %TMi.P 时， %TMi.Q 输出位将设为 0
(4) %TMi.V 等于 %TMi.P 且输入 IN 返回到 0 时，当前值 %TMi.V 将设为 0
(5) 该 Timer 无法复位
(6) 当 %TMi.V 等于 %TMi.P 且输入 IN 为 0 时，则 %TMi.Q 将设为 0

编程示例

简介

Timer 功能块具有下列操作模式：

- TON (Timer On-Delay) (参见第 *133* 页)：用于指定介于激活特定输入与打开某一输出传感器之间的周期。
- TOF (Timer Off-Delay) (参见第 *134* 页)：用于指定介于同某一不再进行检测的传感器相关的输出与相应关闭的输出之间的周期。
- TP (Timer - Pulse) (参见第 *135* 页)：用于创建精确时段的脉冲。

用户可对 Timers 的延迟或脉冲周期进行编程并可在软件内对其进行配置。

编程

本示例为带可逆指令的 Timer 功能块：

梯级	可逆指令
0	BLK %TM0 LD %M0 IN OUT_BLK LD Q ST %Q0.0 END_BLK
1	LD [%TM0.V<400] ST %Q0.1
2	LD [%TM0.V>=400] ST %Q0.2

本示例为带非可逆指令的同一 Timer 功能块：

梯级	非可逆指令
0	LD %M0 IN %TM0
1	LD %TM0.Q ST %Q0.0
2	LD [%TM0.V<400] ST %Q0.1
3	LD [%TM0.V>=400] ST %Q0.2

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

节 4.3
LIFO/FIFO 寄存器 (%R)

使用 LIFO/FIFO 寄存器功能块

本节介绍如何使用 LIFO/FIFO Register 功能块并提供其编程指南。

本节包含了哪些内容？

本节包含了以下主题：

描述

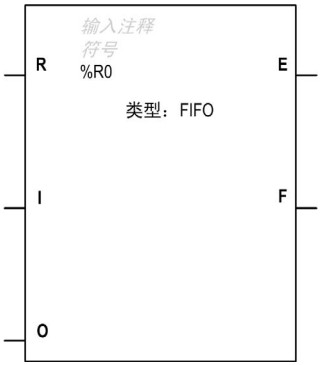
简介

LIFO/FIFO Register 功能块  是一种可通过下列两种不同方式存储多达 16 个 16 位字的存储器块：

- 队列 （先入先出）式，也称为 FIFO。
- 堆栈 （后入先出）式，也称为 LIFO。

示意图

本示意图为 LIFO/FIFO Register 功能块。



输入

LIFO/FIFO Register 功能块具有以下输入：

标签	描述	值
R	复位输入 （或指令）	在状态 1，初始化 LIFO/FIFO Register。
I	存储输入 （或指令）	在上升沿，存储 LIFO/FIFO Register 中关联字 %Ri.I 的内容。
O	检索输入 （或指令）	在上升沿，将 LIFO/FIFO Register 的数据字加载到关联字 %Ri.O 中。

输出

LIFO/FIFO Register 功能块具有以下输出：

标签	描述	值
E	空输出 (%Ri.E)	关联位 %Ri.E 指示 LIFO/FIFO Register 为空。可以测试 %Ri.E 的值，例如，在动态数据表中或使用指令。
F	完整输出 (%Ri.F)	关联位 %Ri.F 指示 LIFO/FIFO Register 已满。可以测试 %Ri.F 的值，例如，在动态数据表中或使用指令。

编程示例

参数

要配置参数，请执行配置功能块过程 (参见第 *129* 页)，并阅读 SoMachine Basic 操作指南中存储器分配模式的描述。

LIFO/FIFO Register 功能块具有以下参数：

参数	描述	值
已使用	已使用的地址	如果选择此参数，则当前在程序中使用此地址。
地址	LIFO/FIFO Register 对象地址	程序只能包含数量有限的 LIFO/FIFO Register 对象。有关寄存器的最大数量，请参阅硬件平台的编程指南。
符号	符号	与此对象关联的符号。有关详细信息，请参阅 SoMachine Basic 操作指南，定义和使用符号。
类型	LIFO/FIFO Register 类型	**FIFO**（队列）或 **LIFO**（堆栈）。
注释	注释	可以将注释与此对象进行关联。

对象

LIFO/FIFO Register 功能块具有以下对象：

对象	描述	值
%Ri.I	LIFO/FIFO Register 输入字	可以读取、测试和写入。 可以在动态数据表中进行修改。
%Ri.O	LIFO/FIFO Register 输出字	可以读取、测试和写入。 可以在动态数据表中进行修改。
%Ri.E	空输出	请参见上表中的“输出”。
%Ri.F	完整输出	请参见上表中的“输出”。

特殊情况

此表包含对 LIFO/FIFO Register 功能块进行编程的特殊情况列表：

特殊情况	描述
冷重启 (%S0=1) 或 **INIT** 的影响	初始化 LIFO/FIFO Register 的内容。与输出 E 关联的输出位 %Ri.E 设置为 1。
热重启 (%S1=1) 或控制器停止的影响	对 LIFO/FIFO Register 的当前值没有任何影响，也不会对其输出位的状态造成影响。

注意：INIT 的影响与 %S0=1 相同。

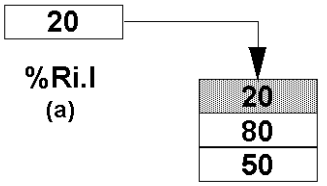
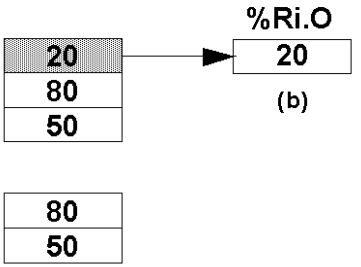
LIFO 寄存器操作

简介

在 LIFO （后入先出）操作中，将首先检索输入的最后一个数据项。

操作

LIFO 操作如下表所示：

阶段	描述	示例
1	**存储**： 收到存储请求时 （输入 I 处的上升沿或指令 I 的激活），输入字 %Ri.I 的内容将存储在栈顶（图 a）的顶部。当堆栈已满时 （输出 F=1 时），则无法再进行存储。	%Ri.I 的内容存储在栈顶。 
2	**检索**： 收到检索请求时 （处于 O 的上升沿或激活指令 O 时），最高数据字 （最后输入的字）将加载到字 %Ri.O 中 （图 b）中。当 LIFO/FIFO Register 为空时 （输出 E=1），将无法再进行检索。输出字 %Ri.O 将不会更改，将暂存其值。	获取栈中最高的数据字。 
3	**结果**： 堆栈可随时复位 （R 处于状态 1 或激活指令 R 时）。指针所指的元素将位于栈内最高处。	–

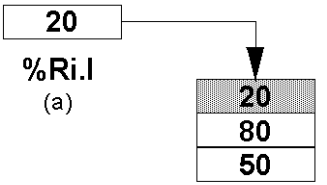
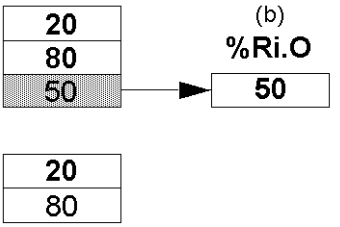
FIFO 寄存器操作

简介

在 FIFO 操作 （先入先出）中，将最先检索输入的第一个数据项。

操作

FIFO 操作如下表所示：

阶段	描述	示例
1	**存储**： 收到存储请求时（输入 I 处的上升沿或指令 I 的激活），输入字 %Ri.I 的内容将存储在队列顶部（图 a）的顶部。当队列已满时 （输出 F=1），则无法再进 行存储。	%Ri.I 的内容存储在队列顶部。 
2	**检索**： 收到检索请求时（处于输入 O 的上升沿或激活指令 O 时），队列中的最低数据字将加载到输出字 %Ri.O 中，并且 LIFO/FIFO Register 的内容将在队列中下移一个位置 （图 b）中下移一个位置。 当 LIFO/FIFO Register 为空时 （输出 E=1），将无法再进行检索。输出字 %Ri.O 将不会更改，将暂存其值。	获取第一个数据项，然后将其载入到 %Ri.O 中。 
3	**结果**： 队列可随时复位（输入 R 处于状态 1 或激活指令 R 时）。	–

编程示例

简介

以下编程示例显示当 LIFO/FIFO Register %R2 未满时 (%R2.F = 0)，在收到存储请求 (%I0.2) 后将加载到 LIFO/FIFO Register (%R2.I) 的存储器字 (%MW34) 的内容。LIFO/FIFO Register 中的存储请求由 %M1 发出。检索请求由输入 %I0.3 确认。如果寄存器不为空 (%R2.E = 0)， %R2.O 将载入 %MW20。

编程

本示例为带可逆指令的 LIFO/FIFO Register 功能块：

梯级	可逆指令
0	BLK %R2 LD %M1 I LD %I0.3 ANDN %R2.E O END_BLK
1	LD %I0.3 [%MW20:=%R2.O]
2	LD %I0.2 ANDN %R2.F [%R2.I:=%MW34] ST %M1

本示例为带非可逆指令的同一 LIFO/FIFO Register 功能块：

梯级	非可逆指令
0	LD %M1 I %R2
1	LD %I0.3 ANDN %R2.E O %R2
2	LD %I0.3 [%MW20:=%R2.O]
3	LD %I0.2 ANDN %R2.F [%R2.I:=%MW34] ST %M1

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

节 4.4
移位寄存器 (%SBR)

使用移位寄存器功能块

本节介绍如何使用 Shift Bit Register 功能块并提供其编程指南。

本节包含了哪些内容?

本节包含了以下主题：

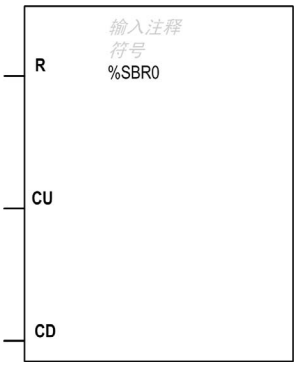
描述

简介

Shift Bit Register 功能块  提供了二进制数据位 （0 或 1）的左移位或右移位。

示意图

本示意图为 Shift Bit Register 功能块：



Shift Bit Register 的当前值显示在功能块的中心：

- 十进制值，如 7
- 二进制值，如 111
- 十六进制值，如 16#7

输入

Shift Bit Register 功能块具有以下输入：

标签	描述	值
R	复位输入 （或指令）	当功能参数 R 为 1 时，这会将寄存器位 0 到 15 %SBRi.j 设置为 0。
CU	移至左输入 （或指令）	在上升沿上，左移寄存器位。
CD	移至右输入 （或指令）	在上升沿上，右移寄存器位。

配置

参数

要配置参数，请执行配置功能块过程 (参见第 *129* 页)，并阅读 SoMachine Basic 操作指南中存储器分配模式的描述。

Shift Bit Register 功能块具有以下参数：

参数	描述	值
已使用	已使用的地址	如果选择此参数，则当前在程序中使用此地址。
地址	**Shift Bit Register** 对象地址	程序只能包含数量有限的 Shift Bit Register 对象。有关寄存器的最大数量，请参阅硬件平台的编程指南。
符号	符号	与此对象关联的符号。有关详细信息，请参阅 SoMachine Basic 操作指南，定义和使用符号。
注释	注释	可以将注释与此对象进行关联。

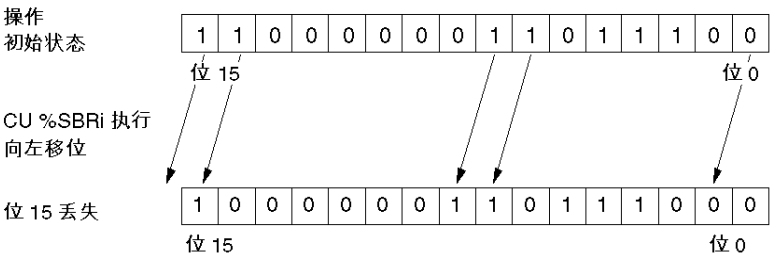
对象

Shift Bit Register 功能块具有以下对象：

对象	描述	值
%SBRi	寄存器编号	0 到 7 可以在动态数据表中进行修改。
%SBRi.j	寄存器位	通过测试指令可以测试移位寄存器的位 0 到 15 （j = 0 到 15），而使用分配指令可以写入位 0 到 15 （j = 0 到 15）。

操作

此图显示了移位操作之前和之后的位模式。



请求使用 CD 指令向右 （位 15 到位 0）移一位也是如此。位 0 丢失。

如果 16 位寄存器不够用，则可以使用该程序层叠若干个寄存器。

特殊情况

此表包含对 Shift Bit Register 功能块进行编程的特殊情况列表：

特殊情况	描述
冷重启 (%S0=1) 的影响	将寄存器字的所有位设为 0。
热重启 (%S1=1) 的影响	对寄存器字的位没有影响。

编程示例

简介

Shift Bit Register 功能块提供了二进制数据位 （0 或 1）的左移位或右移位。

编程

在本例中，当假设位 0 的状态是位 15 时，则每隔一秒钟左移一位。

在可逆指令中：

梯级	可逆指令
0	BLK %SBR0 LD %S6 CU END_BLK
1	LD %SBR0.15 ST %SBR0.0

在非可逆指令中：

梯级	非可逆指令
0	LD %S6 CU %SBR0
1	LD %SBR0.15 ST %SBR0.0

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

节 4.5
步进计数器 (%SC)

使用步进计数器功能模块

本节介绍如何使用 Step Counter 功能块并提供其编程指南。

本节包含了哪些内容？

本节包含了以下主题：

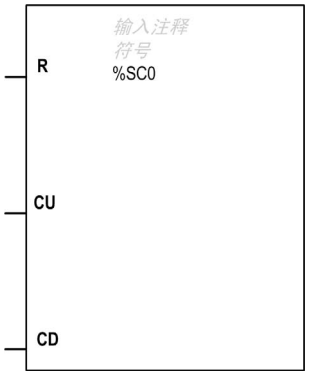
描述

简介

Step Counter 功能块  提供了一系列可以向其分配动作的步进。从一步移动到另一步取决于外部或内部事件。每次激活一步时，均会将关联位 （Step Counter 位 %SCi.j）设置为 1。一个 Step Counter 中每次只能激活一步。

示意图

本示意图为 Step Counter 功能块：



输入

Step Counter 功能块具有以下输入：

标签	描述	值
R	复位输入 （或指令）	当功能参数 R 为 1 时，将复位 Step Counter。
CU	递增输入 （或指令）	在上升沿上，将递增 Step Counter 一步。
CD	递减输入 （或指令）	在上升沿上，将递减 Step Counter 一步。

配置

参数

要配置参数，请执行配置功能块过程 (参见第 *129* 页)，并阅读 SoMachine Basic 操作指南中存储器分配模式的描述。

Step Counter 功能块具有以下参数：

参数	描述	值
已使用	已使用的地址	如果选择此参数，则当前在程序中使用此地址。
地址	Step Counter 对象地址	程序只能包含数量有限的 Step Counter 对象。有关 Step Counter 的最大数量，请参阅硬件平台的编程指南。
符号	符号	与此对象关联的符号。有关详细信息，请参阅 SoMachine Basic 操作指南，定义和使用符号。
注释	注释	可以将注释与此对象进行关联。

对象

Step Counter 功能块具有以下对象：

对象	描述	值
%SCi.j	Step Counter 位	通过载入逻辑运算可以测试 Step Counter 位 0 到 255（j = 0 到 255），而通过赋值指令可以写入位 0 到 255 （j = 0 到 255）。可以在动态数据表中进行修改。

特殊情况

此表包含对 Step Counter 功能块进行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 Step Counter。
热重启 (%S1=1) 的影响	对 Step Counter 没有影响。

编程示例

简介

本示例为 Step Counter 功能块。

- 通过输入 %I0.1 递减 Step Counter 0。
- 通过输入 %I0.2 递增 Step Counter 0。
- 通过输入 %I0.3 或其达到步进 3 时，将 Step Counter 0 复位为 0。
- 步进 0 控制输出 %Q0.1，步进 1 控制输出 %Q0.2，步进 2 控制输出 %Q0.3。

编程

本示例为带可逆指令的 Step Counter 功能块：

梯级	可逆指令
0	BLK %SC0 LD %SC0.3 OR %I0.3 R LD %I0.2 CU LD %I0.1 CD END_BLK
1	LD %SC0.0 ST %Q0.1
2	LD %SC0.1 ST %Q0.2
3	LD %SC0.2 ST %Q0.3

本示例为带非可逆指令的 Step Counter 功能块：

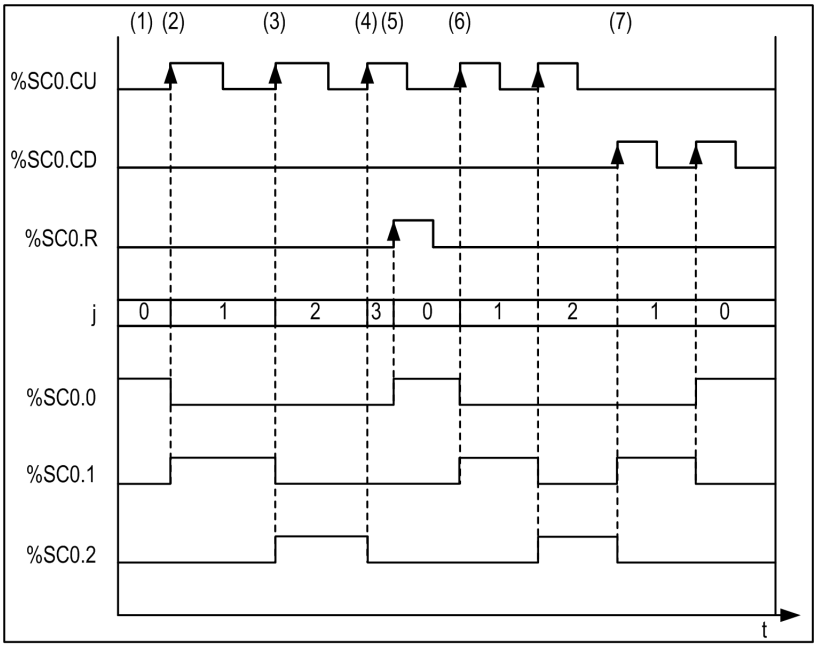
梯级	非可逆指令
0	LD %SC0.3 OR %I0.3 R %SC0
1	LD %I0.2 CU %SC0
2	LD %I0.1 CD %SC0
3	LD %SC0.0 ST %Q0.1

梯级	非可逆指令
4	LD %SC0.1 ST %Q0.2
5	LD %SC0.2 ST %Q0.3

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

时序图

下图说明 Step Counter 功能块的操作：



(1) 步进 0 处于活动状态，因此 %SC0.0 设置为 1
(2) 在 CU 输入的上升沿，将递增步进并更新输出
(3) 递增步进并更新输出
(4) 步进 3 处于活动状态，因此 **Reset** 输入将在一个 CPU 周期后处于活动状态
(5) 当 **Reset** 处于活动状态时，当前步进将设置为 0 且复位输入将在一个 CPU 周期后设置为 0
(6) 当前步进在 CU 输入的上升沿递增
(7) 在 CD 输入的上升沿，将递减步进并更新输出

节 4.6
计数器 (%C)

使用计数器功能块

本节介绍如何使用 Counter 功能块并提供其编程指南。

本节包含了哪些内容?

本节包含了以下主题：

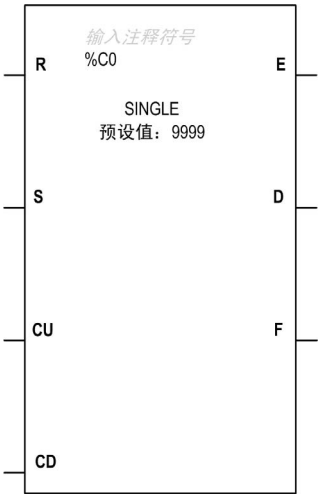
描述

简介

Counter 功能块 **123** 提供了事件的加和减计数。用户可同时执行这两项操作。

示意图

本示意图为 Counter 功能块。



输入

Counter 功能块具有以下输入：

标签	描述	值
R	复位输入 （或指令）	当复位输入 (**R**) 设置为 1 时，将计数器 (%Ci.V) 设置为 0。
S	设置输入 （或指令）	当设置输入 (**S**) 设置为 1 时，将计数器 (%Ci.V) 设置为预设值 (%Ci.P)。
CU	加计数	在计数向上输入 (**CU**) 处，计数器的值在上升沿上 (%Ci.V) 按 1 递增。
CD	减计数	在计数向下输入 (**CD**) 处，计数器的值 (%Ci.V) 在上升沿上按 1 递减。

输出

Counter 功能块具有下列输出：

标签	描述	值
E	减计数溢出	当计数器达到 0 值时，关联位 %Ci.E（计数器为空）将设为 1。出现下列递增情况时，计数器值将达到 9999。
D	已达到预设输出	当 %Ci.V = %Ci.P 时，关联位 %Ci.D（完成计数）将设为 1。
F	加计数溢出	当 %Ci.V 从 9999 变为 0 时，关联位 %Ci.F=1（计数器已满）（%Ci.V 达到 0 时设置为 1；若 Counter 继续执行加计数时，则复位为 0）。

配置

参数

要配置参数，请执行配置功能块过程 (参见第 *129* 页)，并阅读 SoMachine Basic 操作指南中存储器分配模式的描述。

Counter 功能块具有以下参数：

参数	描述	值
已使用	已使用的地址	如果选择此参数，则当前在程序中使用此地址。
地址	Counter 对象地址	程序只能包含数量有限的计数器对象。有关计数器的最大值，请参见控制器的*编程指南*。
符号	符号	与此对象关联的符号。有关详细信息，请参阅 SoMachine Basic 操作指南，定义和使用符号。
预设	预设值	预设值可接受的值 [0 - 9999]。默认值为 9999。使用相关的对象 %Ci.P 可对已配置的此值进行读取、测试和修改。
注释	注释	可以将注释与此对象进行关联。

对象

Counter 功能块具有以下对象：

对象	描述	值
%Ci.V	Counter 的当前值	此字根据输入 （或指令） **CU** 和 **CD** （参见输入表 (参见第 *156* 页)）递增或递减。只能读取。 可以在动态数据表中进行修改。
%Ci.P	预设值	请参见参数表 (参见第 *158* 页)。 可以在动态数据表中进行修改。
%Ci.E	空	请参见输出表 (参见第 *157* 页)。 可以在动态数据表中进行修改。
%Ci.D	完成	请参见输出表 (参见第 *157* 页)。 可以在动态数据表中进行修改。
%Ci.F	完全	请参见输出表 (参见第 *157* 页)。 可以在动态数据表中进行修改。

操作

此表介绍了 Counter 功能块操作的主要阶段：

操作	动作	结果
复位	将输入 R 设置为状态 1 （或激活 R 指令）。	将当前值 %Ci.V 强制设置为 0。将输出 %Ci.E，%Ci.D 和 %Ci.F 设置为 0。复位输入具有优先权。
将	如果将输入 S 设置为 1（或激活 S 指令）并且复位输入为 0 （或 R 指令处于不活动状态）。	当前值 %Ci.V 采用 %Ci.P 值，且将 %Ci.D 输出设置为 1。
计数	上升沿在加计数输入 CU 处显示 （或激活指令 CU）。	%Ci.V 当前值增加一个单位。
	%Ci.V 当前值等于 %Ci.P 预设值。	将 " 已达到预设值 " 输出位 %Ci.D 切换为 1。
	将 %Ci.V 当前值从 9999 更改为 0。	将输出位 %Ci.F （加计数溢出）切换为 1。
	如果 Counter 继续加计数。	将输出位 %Ci.F （加计数溢出）复位为 1。
减计数	上升沿在减计数输入 CD 处显示 （或激活 CD 指令）。	当前值 %Ci.V 减少一个单位。
	将当前值 %Ci.V 从 0 更改为 9999。	将输出位 %Ci.E （减计数溢出）切换为 1。
	如果 Counter 继续减计数。	将输出位 %Ci.F （减计数溢出）复位为 0。

特殊情况

此表显示 Counter 功能块的特殊操作 / 配置情况列表：

特殊情况	描述
冷重启 (%S0=1) 或 **INIT** 的影响	● 将当前值 %Ci.V 设置为 0。 ● 将输出位 %Ci.E，%Ci.D，和 %Ci.F 设置为 0。 ● 在配置的过程中，使用定义的值初始化预设值。
控制器停止的热重启 (%S1=1) 的影响	对 Counter (%Ci.V) 的当前值没有任何影响。
修改预设 %Ci.P 的影响	由应用程序对块进行处理 （激活其中的一个输入）时，通过指令或通过调整它修改预设值才会生效。

注意：INIT 的影响与 %S0=1 相同。

编程示例

简介

以下示例为最多可提供 5,000 个项目的计数器。输入 %I0.2 上的每个脉冲（当内存位 %M0 设置为 1 时）都会使 Counter 功能块 %C8 发生递增，直至达到其最终预设值 （位 %C8.D=1）。该计数器将由输入 %I0.1 进行复位。

编程

本示例为带可逆指令的 Counter 功能块：

梯级	可逆指令
0	BLK %C8 LD %I0.1 R LD %I0.2 AND %M0 CU END_BLK
1	LD %C8.D ST %Q0.0

本示例为带非可逆指令的同一 Counter 功能块：

梯级	非可逆指令
0	LD %I0.1 R %C8
1	LD %I0.2 AND %M0 CU %C8
2	LD %C8.D ST %Q0.0

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

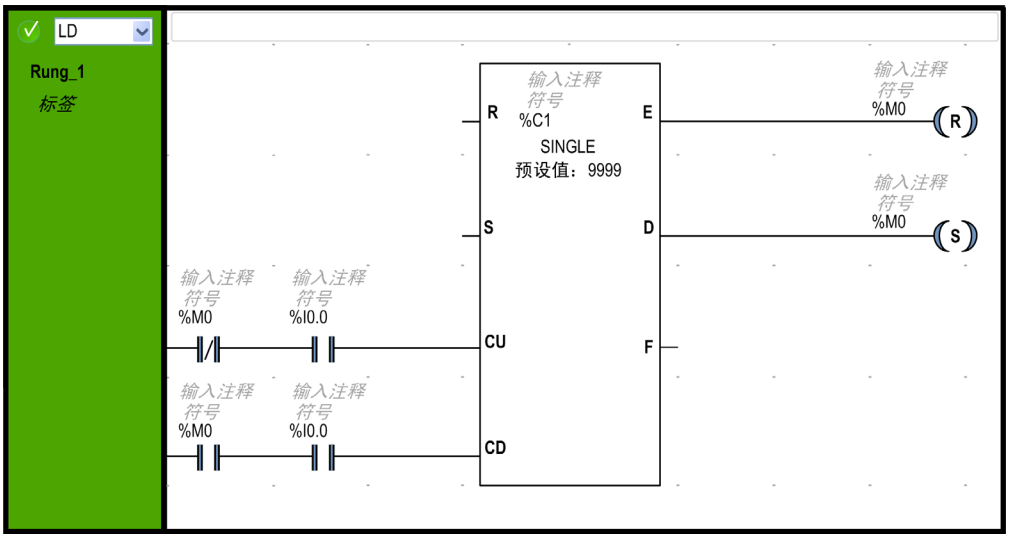
配置

配置过程中，须输入下列参数：

预设值 (%Ci.P)：在本例中设置为 5000。

加 / 减计数器示例

本示意图为 Counter 功能块的示例。



在本例中， %M0 为递增 (%M0 = False) 和递减 (%M0 = True) 次序。计数器将对 %I0.0 的前沿进行计数。如果 %M0 为 False，则在 %I0.0 上的每个前沿上， %C1.V 发生递增，直至达到预设的 %C1.P 值，同时 Done 指示器 %C1.D 切换为 True。此外， %C1.D 输出将设置 %M0 并将指令切换为递减次序。然后在 %I0.0 的前沿上， %C1.V 发生递减，直至达到 0。 Empty 指示器 (%C1.E) 开启，并重置 %M0 （增量顺序）。

节 4.7
快速计数器 (%FC)

高速计数器

概述

请参阅 Controller 的*编程指南*。

节 4.8
高速计数器 (%HSC)

高速计数器

概述

请参阅 Controller 的*编程指南*。

节 4.9
Drum 寄存器 (%DR)

使用 Drum 寄存器功能块

本节介绍如何使用 Drum Register 功能块并提供其编程指南。

本节包含了哪些内容？

本节包含了以下主题：

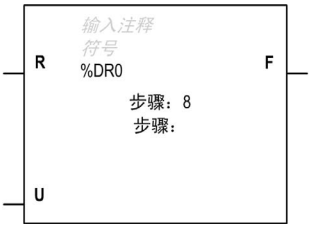
描述

简介

Drum Register 功能块  的操作原理类似于机电 Drum Register，根据外部事件更改步骤。凸轮的高点在每个步进都会提供由控制器执行的命令。如果有 Drum Register 功能块，这些高点通过每个步进的状态 1 用符号表示，并且分配给输出位 %Qi.j 或内存位 %Mi。

示意图

本示意图为离线模式下的 Drum Register 功能块。



步骤 显示在 **Drum 助手**中配置的步进总数。
步进 创建块时在离线模式下显示。在在线模式下，则显示当前步进编号。

输入

Drum Register 功能块具有以下输入：

标签	描述	值
R	返回到步进 0 （或指令）	在状态 1，将 **Drum Register** 设置为步进 0。
U	前进输入 （或指令）	在上升沿，使 **Drum Register** 前进 1 步并更新控制位。

输出

Drum Register 功能块具有以下输出：

标签	描述	值
F	输出 (%DRi.F)	表明当前步进与定义的最后步进相等。可对关联位 %DRi.F 进行测试。

配置

参数

要配置参数，请执行配置功能块过程 (参见第 *129* 页)，并阅读 SoMachine Basic 操作指南中存储器分配模式的描述。

Drum Register 功能块具有以下参数：

参数	描述	值
已使用	已使用的地址	如果选择此参数，则当前在程序中使用此地址。
地址	Drum Register 对象地址	程序只能包含数量有限的 Drum Register 对象。有关 Drum Register 的最大值，请参见控制器的*编程指南*。
符号	符号	与此对象关联的符号。有关详细信息，请参阅 SoMachine Basic 操作指南，定义和使用符号。
配置	Drum 助手	**步进数**：1...8。 与步骤关联的输出位或内存位：**位 0 ... 位 15**。
注释	注释	可以将注释与此对象进行关联。

对象

Drum Register 功能块具有以下对象：

对象	描述	值
%DRi.S	当前步进编号	**0<=**%DRi.S**<=7**。可以读取和写入的字。写入值必须为十进制立即值。写入时，将影响功能块的下一步执行。 可以在动态数据表中进行修改。
%DRi.F	完全	请参见输出表 (参见第 *165* 页)。

操作

Drum Register 功能块包含：

- 在八步 （0 到 7）内组织的常量数据 （凸轮）矩阵和在编号为 0 到 15 的列中排列的 16 个数据位 （步进的状态）。
- 与配置输出 (%Qi.j) 或存储器字 (%Mi) 关联的控制位列表。在当前步进过程中，控制位采用为此步进定义的二进制状态。

本示例概括了 Drum Register 的主要特性：

注意：也可以在内存位进行配置 (%Mi)。

梯级	指令
0	BLK %DR0 LD %M10 R LD %M11 U END_BLK

在动态数据表中创建以下条目：%M10, %M11, %DR0, %Q0.0 至 %Q0.5。

当 %M11 的值 （DRUM 的演变）或 %M10 的值 （DRUM 的复位）发生变化时，请观察 %DR0.S, %DR0.F, %Q0.0 至 %Q0.5 的值的变化。然后，再看看溢出情况 （返回到步骤 0）。

特殊情况

此表包含 Drum Register 操作的特殊情况列表。

特殊情况	描述
冷重启 (%S0=1) 的影响	将 Drum Register 复位为步 0 （更新控制位）。
热重启 (%S1=1) 的影响	在当前步骤之后更新控制位。
程序跳转的影响	不再扫描不再扫描 Drum Register 意味着控件位保持其上一次的状态。
更新控制位	仅在更改步骤或进行热重启或冷重启的情况下发生。

编程示例

简介

下面是对已配置的 Drum Register 进行编程的示例，这样所有控件都不会在步骤 0 设置，并且在输出 %Q0.0 至 %Q0.5 上时，分别在步骤 1 至步骤 6 设置控件 （参见配置 (参见第 *171* 页)）。

每当输入 %I0.1 设置为 1 时，前 6 个输出（%Q0.0 到 %Q0.5）便会依次激活。当输入 %I0.0 较高时，它会将以下值重置为 0。

- Drum Register 输出 **F** (%DRi.F = 0)
- 当前步骤编号 (%DRi.S = 0)

编程

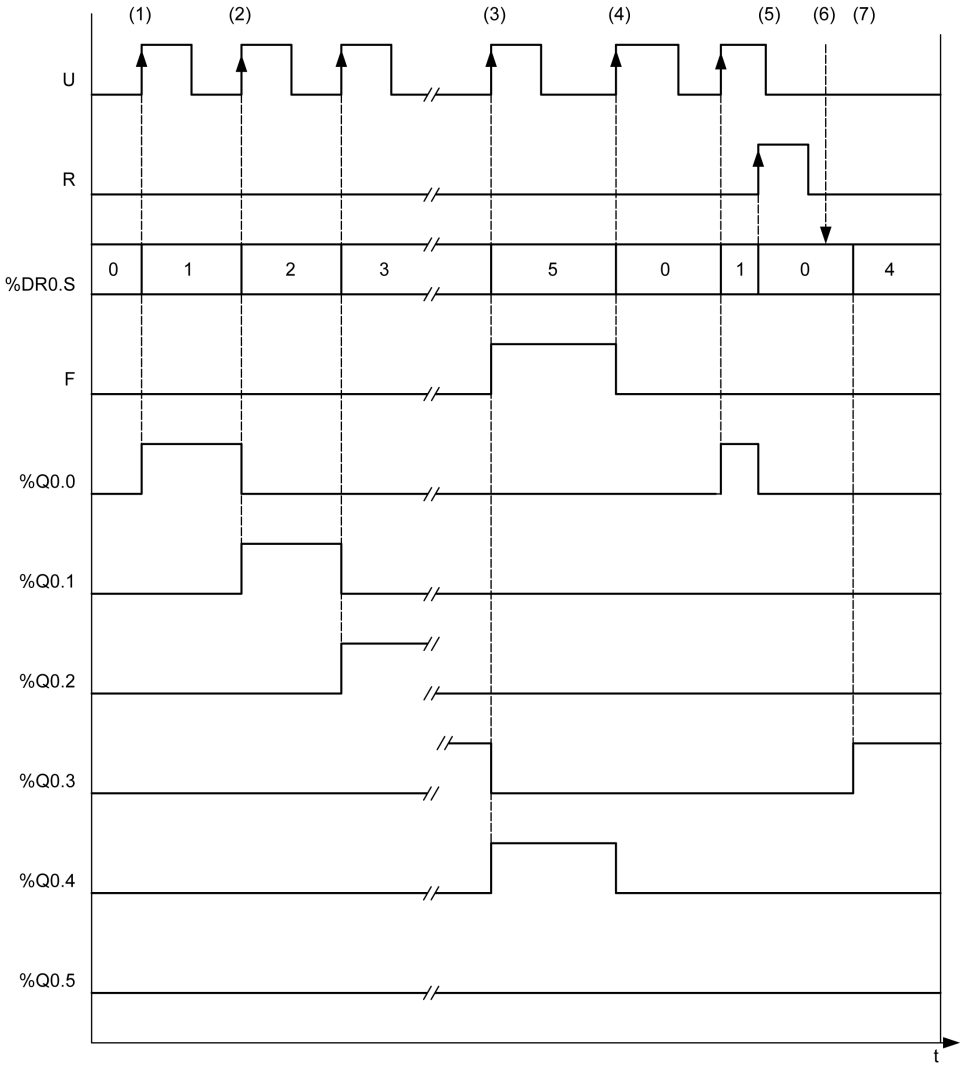
本示例为 Drum Register 功能块程序：

梯级	指令
0	BLK %DR1 LD %I0.0 R LD %I0.1 U OUT_BLK LD F ST %Q0.7 END_BLK

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

时序图

下图说明 Drum Register 的操作：



(1) 在 U 输入的上升沿，当前步进将递增
(2) 当前步骤更新时，输出也将相应更新
(3) 到达最后步进时，输出 F 将设置为 1
(4) 当最后步进处于活动状态且输入 U 处于上升沿时，当前步进将复位为 0
(5) %DR0.R = 1（上升沿）当前值将设置为 0

(6) 由用户记录步进编号的值：`%DR0.S = 4`
(7) 用户所记录的值将在下次执行时更新

配置

以下信息在配置过程中定义：

- 步进数：6
- 每个 `Drum Register` 步进的输出状态 （控制位）：

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

- 控制位的分配：

控制位的关联输出如下表所示：

位	关联输出
0	无关联输出
1	`%Q0.1`
2	`%Q0.2`
3	`%Q0.3`
4	`%Q0.4`
5	`%Q0.5`

节 4.10
脉冲 (%PLS)

脉冲

概述

请参阅 Logic Controller 的*编程指南*。

节 4.11
脉冲宽度调制 (%PWM)

脉冲宽度调制

概述

请参阅 Logic Controller 的*编程指南*。

节 4.12
消息 (%MSG) 和交换 (EXCH)

使用消息功能块

本节介绍如何使用 Message 功能块并提供其编程指南。

本节包含了哪些内容？

本节包含了以下主题：

概述

简介

Logic Controller 可以配置为通过 Modbus 协议进行通讯，也可以在字符模式 (ASCII) 下发送和 / 或接收消息。

SoMachine Basic 提供了以下功能来进行这些通讯：

- **交换** (EXCH) 指令，用于传输 / 接收消息。
- Message 功能块 (%MSG)，用于控制数据交换。

Logic Controller 使用在处理**交换**指令时为指定端口配置的协议。可以给各通讯端口分配不同的协议。通过将端口号附加到**交换**指令 （EXCH1、 EXCH2）或 Message 功能块 （%MSG1、 %MSG2）来访问通讯端口。

Logic Controller 使用 EXCH3 指令和 %MSG3 功能块，通过以太网网络实现 Modbus TCP 消息传递。

下表显示了用于访问控制器通讯端口的**交换**指令和 Message 功能块：

通讯端口	**交换指令**	**Message 功能块**
2 路串行线路	EXCH1	%MSG1
	EXCH2	%MSG2
1 路串行线路和 1 以太网	EXCH1	%MSG1
	EXCH3	%MSG3

交换指令

交换指令可让 Logic Controller 将信息发送到 ASCII 或 Modbus 设备和 / 或接收来自 ASCII 或 Modbus 设备的信息。定义一个字表 (%MWi:L)，该表包含控制信息以及所发送和 / 或所接收的数据。请参阅配置传输表 (参见第 *181* 页)。消息交换是使用**交换**指令来执行的。

语法

下面是**交换**指令的格式：

```
[EXCHx %MWi:L]
```

其中 x 是端口号； L 是字表的总字数。

Logic Controller 必须先完成来自第一个**交换**指令的交换，然后才能开始第二个**交换**指令。发送多条消息时，必须使用 Message 功能块。

ASCII 协议

ASCII 协议为逻辑控制器提供了一种简单的字符模式协议，以便使用简单的设备来传输和 / 或接收数据。此协议使用**交换**指令进行支持，并使用 Message 功能块进行控制。

使用 ASCII 协议可以进行 3 类通讯：

- 仅传输
- 传输 / 接收
- 仅接收

Modbus 协议

对于串行链路， Modbus 协议是一种主站 - 从站协议，它允许一个且仅允许一个主站请求来自从站的响应，或根据请求执行操作。在有以太网支持的情况下，多个主站 （客户端）可与一个从站（服务器）进行交换。每个从站必须具有唯一的地址。主站可以对单个从站进行寻址，也可以向所有从站发出广播消息。从站将向单独发送给它们的查询返回消息 （响应）。不向来自主站的广播查询返回响应。

Modbus 主模式使控制器可以向从站发送 Modbus 查询并等待响应。只能通过**交换**指令来支持 Modbus 主模式。 Modbus 主模式支持 Modbus ASCII 和 RTU。

Modbus 从站模式使控制器可以响应来自 Modbus 主站的标准 Modbus 查询。

有关 Modbus 协议的详细信息，请参阅 *Modbus 应用程序协议*文档，其网址为 *http://www.modbus.org*。

Modbus 从站

Modbus 协议支持 OSI 型号格式的 2 个数据链路层：ASCII 和 RTU。每种格式都由物理层实现定义，包括使用 7 个数据位的 ASCII 格式和使用 8 个数据位的 RTU 格式。

使用 Modbus ASCII 模式时，消息中的每个字节都作为 2 个 ASCII 字符发送。Modbus ASCII 帧以开始字符 （":"）开始，并以 2 个结束字符 （CR 和 LF）结束。默认情况下，帧末字符为 0x0A (LF)。 Modbus ASCII 帧的检查值是帧的简单二进制补码，不包括开始和结束字符。

Modbus RTU 模式不会在传输之前重新设置消息的格式；但是，它将使用指定为 CRC 的不同校验和计算模式。

Modbus 数据链路层具有以下限制：

- 地址 1-247
- 位数：请求时为 128 位
- 字数：请求时为 125 个 16 位的字

描述

简介

Message 功能块  管理数据交换并具有 3 个功能：

- 通讯错误检查：
  错误检查验证各个**交换**表的大小，并验证配置相关交换的有效性。
- 多个消息的协调：
  为确保发送多个消息时的协调一致， Message 功能块会提供所需信息，以确定上一条消息何时完成。
- 优先消息的传输：
  Message 功能块允许停止当前消息传输，以便立即发送紧急消息。

Message 功能块的编程可选。

当检测到错误时，对于交换块 EXCH1、 EXCH2 和 EXCH3，这些错误代码将分别写入系统字 %SW63、 %SW64 和 %SW65。有关详细信息，请参阅控制器的*编程指南*。

示意图

本示意图为 Message 功能块：

输入

Message 功能块具有以下输入：

标签	描述	值
R	复位输入	状态 1：重新初始化通讯； %MSGx.E = 0 和 %MSGx.D = 1。 状态 0：正常模式。

输出

Message 功能块具有以下输出：

标签	描述	值
D	通讯完成 (%MSGx.D	状态 1： ● 传输结束 （如果传输） ● 接收结束 （收到结束字符） ● 错误 ● 复位块 状态 0：请求正在执行。
E	检测到通讯错误 (%MSGx.E)	状态 1： ● 未定义的命令 ● 表配置不正确 ● 收到错误字符 （速度、奇偶校验等） ● 接收表已满 （未更新） 状态 0：消息长度正确，链接已建立。 当检测到通讯错误时，请参阅下表查看写入系统字的错误代码。

通讯错误代码

下表介绍了当检测出通讯错误时写入系统字的错误代码：

系统字	功能	描述
%SW63	EXCH1 功能块错误代码	EXCH1 错误代码： 0 - 操作已成功 1 – 要传输的字节数太多 (> 250) 2 - 传输表太小 3 - 字表太小 4 - 接收表溢出 5 - 已超时 6 - 传输 7 - 表中存在错误命令 8 - 所选端口未配置 / 不可用 9 - 接收错误：此错误代码表示接收帧出现错误。这可能是由于物理参数配置错误 （例如校验位、数据位、波特率等），或者是由于物理连接错误导致信号减弱而引起的。 10 - 接收时不能使用 %KW 11 - 传输偏移大于传输表 12 - 接收偏移大于接收表 13 - 控制器停止了 EXCH 处理
%SW64	EXCH2 功能块错误代码	EXCH2 错误代码：请参见 %SW63。

系统字	功能	描述
%SW65	EXCH3 功能块错误代码	1-4、6-13：请参见 %SW63。（请注意，错误代码 5 无效，由下文介绍的以太网特定的错误代码 109 和 122 取代。） 以下为以太网特定的错误代码： 101 - 无此 IP 地址 102 - TCP 连接断开 103 - 无可用的套接字 （所有连接通道均忙碌） 104 - 网络已关闭 105 - 无法访问网络 106 - 复位时网络丢失连接 107 - 对等设备中止连接 108 - 对等设备复位连接 109 - 连接已超时 110 - 连接尝试被拒绝 111 - 主机已关闭 120 - 未知索引 （配置表中未包含远程设备的索引） 121 - 重大错误 （MAC、芯片、 IP 重复） 122 - 发出数据后接收流程超时 123 - 正在初始化以太网

配置

检测到错误

如果在使用**交换**指令时检测出错误，则将位 %MSGx.D 和 %MSGx.E 设置为 1，且系统字 %SW63 包含端口 1 的错误代码，以及 %SW64 包含端口 2 的错误代码。请参阅 Logic Controller 功能库指南的“系统字”一章。

操作

下表介绍了 Message 功能块操作的主要阶段：

操作	动作	结果
复位	将输入 R 设置为状态 1（或激活 R 指令）。	• 正在传输的所有消息均已停止。 • 将通讯错误输出复位为 0。 • 将完成位设置为 1。 现在，可以发送新消息了。
通讯完成	将输出 D 设置为状态 1。	Logic Controller 准备发送另一条消息。在发送多条消息时建议使用 %MSGx.D 位。如果不使用该位，消息可能会丢失。
检测到通讯错误	将通讯错误输出设置为 1： • 由于通讯编程错误或消息传输错误。 • 如果在与**交换**指令 （字 1，最低有效字节）关联的数据块中定义的字节数超过 128（FA 提供的十六进制值 +80）。 • 如果在将 Modbus 消息发送到 Modbus 设备的过程中存在问题。在这种情况下，您应检查接线，并检查目标设备是否支持 Modbus 通讯。	

特殊情况

此表包含 Message 操作的特殊情况列表。

特殊情况	描述
冷重启 (%S0=1) 或 **INIT** 的影响	强制执行通讯的重新初始化。
热重启 (%S1=1) 的影响	无影响。
控制器停止的影响	如果正在传输消息，则控制器会停止其传输并重新初始化输出 %MSGx.D 和 %MSGx.E。

注意：INIT 的影响与 %S0=1 相同。

限制

请注意以下限制：

- 只能在通电或重置时检查端口 2 （对于 ASCII 协议）的可用性和类型 （请参阅 %SW7）
- 在通电或重置时检查端口 2 （对于 Modbus 协议）的存在和配置 (RS-485)
- 连接 SoMachine Basic 时，端口 1 上的任何消息处理都将中止
- **交换**指令将中止活动的 Modbus 从站处理
- 如果检测到错误，则不重试处理**交换**指令
- 可以使用复位输入 (R) 来中止**交换**指令接收处理
- 在 Modbus 协议的情况下，可以使用超时来配置**交换**指令。
- 通过 %MSGx.D 可控制多条消息

配置传输 / 接收表

传输和 / 或接收帧的最大大小为：

- 250 个字节 （对于 Modbus 协议）。
- 256 个字节 （对于 ASCII 协议）。

与**交换**指令关联的字表包括控制表、传输表和接收表：

	最高有效字节		**最低有效字节**	
	Modbus	**ASCII**	**Modbus**	**ASCII**
控制表	命令		长度 （传输 / 接收）	
	Rx 偏移	保留 (0)	Tx 偏移	保留 (0)
传输表	传输的字节 1		传输的字节 2	
	...		...	
			传输的字节 n	
	传输的字节 n+1			
接收表	接收的字节 1		接收的字节 2	
	...		...	
			接收的字节 p	
	接收的字节 p+1			

注意：除对单个从站进行查询外， Modbus 主控制器还可以向所有从站发送广播查询。必须将广播查询中的**命令**字节设置为 00，同时必须将从站地址设置为 0。

ASCII 协议的控制表

长度字节包含以字节表示的传输表长度 （最大 250 个字节），如果需要接收，则其将在接收结束时被收到的字符数覆盖。

命令字节必须包含以下内容之一：

- 0：仅传输
- 1：发送 / 接收
- 2：仅接收

Modbus 协议的控制表

长度字节包含以字节表示的传输表长度 （最大 250 个字节），如果需要接收，则其将在接收结束时被收到的字符数覆盖。

该参数为以字节表示的传输表长度。如果 **Tx 偏移**参数等于 0，则此参数将等于传输帧的长度。如果 **Tx 偏移**参数不等于 0，则传输表中将有一个字节 （由偏移值指定）不被传输并且此参数等于帧长度加 1。

对于 Modbus RTU 请求 （广播除外），**命令**字节必须始终等于 1 （**Tx** 和 **Rx**）。对于广播，必须为 0。

Tx 偏移字节包含在传输字节时将要忽略的字节在传输表中的序号 （1 表示第一个字节、 2 表示第二个字节，依次类推）。这用于处理 Modbus 协议中与字节 / 字值关联的问题。例如，如果此字节包含 3，则将忽略第三个字节，而使数据表中的第四个字节成为要传输的第三个字节。

Rx 偏移字节包含在传输数据包时将要添加的字节在接收表中的序号 （1 表示第一个字节、 2 表示第二个字节，依次类推）。这用于处理 Modbus 协议中与字节 / 字值关联的问题。例如，如果此字节包含 3，则表中的第三个字节将填充为 0，并且将实际接收的第三个字节输入到表中的第四个位置。

ASCII 协议的传输 / 接收表

在仅传输模式下，将在执行**交换**指令之前填充控制表和传输表，并且可以为 %KW 或 %MW 类型。在仅传输模式下，字符的接收无需空格。在传输所有字节后，将 %MSGx.D 设置为 1，并且可以执行新的**交换**指令。

在传输 / 接收模式下，将在执行**交换**指令之前填充控制表和传输表，并且必须为 %MW 类型。在传输表的结束处需要最多 256 个接收字节的空间。在传输所有字节后，将 Logic Controller 切换到接收模式，并等待接收任何字节。

在仅接收模式下，将在执行**交换**指令之前填充控制表，并且必须为 %MW 类型。在控制表的结束处需要最多 256 个接收字节的空格。 Logic Controller 立即进入接收模式，并等待接收所有字节。

在已收到使用的帧字节结尾或接收表已满时接收结束。在这种情况下，在系统字 %SW63 和 %SW64 中将出现检测到的错误代码 （接收表溢出）。如果已配置非零超时，则当超时完成时接收结束。如果已选择零超时值，将不存在接收超时。因此，要停止接收，必须激活 %MSGx.R 输入。

Modbus 协议的传输 / 接收表

使用某种模式（Modbus ASCII 或 Modbus RTU）时，将在执行**交换**指令之前用请求填充传输表。在执行时，Logic Controller 将确定数据链路层，并且执行处理传输和响应时必需的所有转换。开始字符、结束字符和检查字符不存储在传输表 / 接收表中。

在传输所有字节后，将 Logic Controller 切换到接收模式，并等待接收任何字节。

将以如下几种方式之一完成接收：

- 在字符或帧上检测到超时，
- 在 ASCII 模式中接收到帧字符的结束符，
- 接收表已满。

传输的字节 x 条目包含要传输的 Modbus 协议（RTU 编码）。如果通讯端口配置用于 Modbus ASCII，则将校正帧字符附加到传输。第一个字节包含设备地址（特定地址或广播地址），第二个字节包含功能代码，而其余字节包含与该功能代码关联的信息。

注意：这是典型的应用，但是未定义所有的可能性。将不对传输的数据执行任何验证。

接收的字节 x 条目包含要接收的 Modbus 协议（RTU 编码）。如果通讯端口配置用于 Modbus ASCII，则将校正帧字符从响应中删除。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），而其余字节包含与该功能代码关联的信息。

注意：这是典型的应用，但是未定义所有的可能性。将不对接收的数据执行任何验证（校验和验证除外）。

编程示例

简介

以下是 Message 功能块的编程示例。

对传输多个连续的消息进行编程

在应用程序中，执行**交换**指令将激活 Message 功能块。如果 Message 功能块尚未处于活动状态 (%MSGx.D = 1)，则传输消息。如果在同一周期中发送多条消息，则只有第一条消息使用相同的端口进行传输。

在端口 1 上连续传输两个消息的示例：

梯级	可逆指令	注释
0	LD %M142 [%MW2:=16#0106] [%MW3:=0] [%MW4:=16#0106] [%MW5:=4] [%MW6:=7]	在从站上写入，在地址 1：寄存器 4 上的值 7。 [%MW2:=16#0106]：命令代码：01（十六进制），传输长度：06（十六进制） [%MW3:=0]：无接收或传输偏移 [%MW4:=16#0106]：从站地址：01（十六进制），功能代码：06（十六进制）(写入单个寄存器) [%MW5:=4]：寄存器地址 [%MW6:=7]：要写入的值
1	LD %MSG2.D AND %M0 [EXCH2%MW2:5] R %M0	%MSG2.D：检测端口是否正忙，从而协调多条消息。
2	LDR %I0.0 AND %MSG2.D [EXCH2%MW2:5] S %M0	–

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

对重新初始化交换进行编程

可通过激活输入（或指令）R 取消交换。此输入将初始化通讯，并将输出 %MSGx.E 复位为 0，将输出 %MSGx.D 复位为 1。如果检测到错误，则可以重新初始化交换。

重新初始化交换的示例：

梯级	可逆指令	注释
0	BLK %MSG1 LD %M0 R END_BLK	–

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

ASCII 示例

编写应用程序

ASCII 应用程序示例：

梯级	指令	注释
0	LD 1 [%MW10:=16#0104] [%MW11:=16#0000] [%MW12:=16#4F4B]	[%MW10:=16#0104]：命令代码：01（十六进制），传输长度：04（十六进制） [%MW11:=16#0000]：0000：空 [%MW12:=16#4F4B]：良好 **注意：**下表有 8 个元素。
1	LD 1 AND %MSG2.D [EXCH2 %MW10:8]	
2	LD %MSG2.E ST %Q0.0 END	

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

使用 SoMachine Basic 创建一个三梯级程序：

- 首先，初始化控制和传输以用于**交换**指令。在此示例中，设置了发送和接收数据的命令。要发送的数据量设置为 4 字节，如应用程序中所定义，然后是配置中声明的帧字符的结束。动态数据表中不显示开始和结束字符，而仅显示数据字符。在所有情况下，在使用时，将会在接收（通过 %SW63 和 %SW64）这些字符时自动进行传输或检查。
  注意：在帧的结尾，将自动发送配置中定义的结束字符。例如，如果您已将首个结束字符配置为 10 并将第二个结束字符配置为 13，则在帧结尾发送 16#0A0D（ASCII 代码， 0A = LF 且 0D = CR）。
- 接下来，检查与 %MSG2 关联的状态位，并仅在端口准备就绪时发出 EXCH2 指令。对于 EXCH2 指令，已指定了一个 8 个字的值。有 2 个控制字 （%MW10 和 %MW11）、2 个用于传输信息的字（%MW12 和 %MW13）和 4 个用于接收数据的字 （%MW14 到 %MW17）。
- 最后，将在本地基板控制器 I/O 的第一个输出位上感测和存储检测到的 %MSG2 错误状态。也可以添加使用 %SW64 进行的其他错误处理过程，以使其更精确。

动态数据表初始化

在在线模式下初始化动态数据表的示例：

地址	值	格式
%MW10	0104	十六进制
%MW11	0000	十六进制
%MW12	4F4B	十六进制
%MW13	0A0D	十六进制
%MW14	AL	ASCII
%MW15	OH	ASCII
%MW16	A	ASCII

要显示可能的格式，右键单击动态数据表中的**值**框。

最后的步骤是将此应用程序下载到控制器并运行。初始化动态数据表以激活和显示 %MW10 至 %MW16 的字。可使用 Logic Controller 交换此信息，并显示在动态数据表中。

Modbus 标准请求和示例

Modbus 主站：读取 N 个位

下表表示请求 01 和 02 （对于输出位或内存位为 01，对于输入位为 02）：

	表 索引	最高有效字节	最低有效字节
控制表	0	01 （传输 / 接收）	06 （传输长度）(1)
	1	03 （接收偏移）	00 （传输偏移）
传输表	2	从站 @(1...247)	01 或 02 （请求代码）
	3	在从站中要读取的第一个位的地址	
	4	N_1 = 要读取的位数	
接收表 （响应后）	5	从站 @(1...247)	01 或 02 （响应代码）
	6	00 （由 Rx 偏移操作添加的字节）	N_2 = 要读取的数据字节数 = [1+(N_1-1)/8] 其中结果为除法的整数部分。
	7	第一位的值 （值 00 或 01），扩展到某个字节	第二位的值 （如果 N_2 > 1），扩展到某个字节
	8	第三位的值 （如果 N_1 > 1），扩展到某个字节	–
	...	...	...
	(N_2/2)+6 （如果 N_2 为偶数） (N_2/2+1)+6 （如果 N_2 为奇数）)	第 N_2 位的值（如果 N_1> 1 ），扩展到某个字节	–
(1) 该字节同样接收响应后传输的字符串长度。			

Modbus 主站：读取 N 个字

下表表示请求 03 和 04 （对于输出字或内存字为 03，对于输入字为 04）：

	表 索引	最高有效字节	最低有效字节
控制表	0	01 （传输 / 接收）	06 （传输长度） (1)
	1	03 （接收偏移）	00 （传输偏移）
传输表	2	从站 @(1...247)	03 或 04 （请求代码）
	3	要读取的第一个字的地址	
	4	N = 要读取的字数	
接收表 （响应后）	5	从站 @(1...247)	03 或 04 （响应代码）
	6	00 （由 Rx 偏移操作添加的字节）	2*N （读取的字节数）
	7	读取的第一个字	
	8	读取的第二个字 （如果 N>1）	
	...	...	
	N+6	读取的第 N 个字 （如果 N>2）	
(1) 该字节同样接收响应后传输的字符串长度。			

注意：以上三处接收偏移均将在接收表中的第三个位置添加一个字节 （值 = 0）。这样可以确保能够对在该数据表中要读取的字节数和字的值进行方便的定位。

Modbus 主站：写入位

下表表示请求 05 （写入单个位：输出或内存）：

	表 索引	最高有效字节	最低有效字节
控制表	0	01 （传输 / 接收）	06 （传输长度） (1)
	1	00 （接收偏移）	00 （传输偏移）
传输表	2	从站 @(1...247) 如果是广播，则为 0	05 （请求代码）
	3	要为索引字 4 的 MSB 写入的值； 0xFF 或 0x00 (2)。	
	4	在从站中要写入的位值 （16#0000 = False， 16#FF00 = True）	
接收表 （响应后）	5	从站 @(1...247)	05 （响应代码）
	6	已写入的位的地址	
	7	已写入的值	
(1) 该字节同样接收响应后传输的字符串长度。 **(2)** 对于要写入 1 的位，传输表中的关联字必须包含值 FF00h，对于要写入 0 的位必须包含值 0。			

注意：

- 该请求无需使用偏移。
- 此处响应帧与请求帧相同 （正常情况下）。

Modbus 主站：写入字

下表表示请求 06 （写入单个字：输出或内存）：

	表 索引	最高有效字节	最低有效字节
控制表	0	01 （传输 / 接收）	06 （传输长度）(1)
	1	00 （接收偏移）	00 （传输偏移）
传输表	2	从站 @(1...247) 如果是广播，则为 0	06 （请求代码）
	3	要写入的字的地址	
	4	要写入字的值	
接收表 （响应后）	5	从站 @(1...247)	06 （响应代码）
	6	已写入字的地址	
	7	已写入的值	
(1) 该字节同样接收响应后传输的字符串长度。			

注意：

- 该请求无需使用偏移。
- 此处响应帧与请求帧相同 （正常情况下）。

Modbus 主站：写入 N 个位

下表表示请求 15 （写入 N 个位：输出或内存）：

	表 索引	最高有效字节	最低有效字节
控制表	0	01 （传输 / 接收）	8 + 字节数 （传输）
	1	00 （接收偏移）	07 （传输偏移）
传输表	2	从站 @(1...247) 如果是广播，则为 0	15 （请求代码）
	3	要写入的第一个字的地址	
	4	N_1 = 要写入的位数	
	5	00 （未发送的字节，偏移影响）	N_2 = 要写入的数据字节数 = [1+(N_1-1)/8] 其中结果为除法的整数部分。
	6	第一个字节的值	第二个字节的值
	7	第三个字节的值	第四个字节的值
	...	...	...
	(N_2/2)+5 （如果 N_2 为偶数） (N_2/2+1)+5 （如果 N_2 为奇数）)	第 N_2 个字节的值	
接收表 （响应后）	–	从站 @(1...247)	15 （响应代码）
	–	已写入的第一个位的地址	
	–	已写入的位数 (= N_1)	

注意： 传输偏移 = 7 表示抑制已发送帧中的第七个字节。这还可以使传输表中字的值进行良好的通讯。

Modbus 主站：写入 N 个字

下表表示请求 16：

	表 索引	最高有效字节	最低有效字节
控制表	0	01 （传输 / 接收）	8 + (2*N) （传输长度）
	1	00 （接收偏移）	07 （传输偏移）
传输表	2	从站 @(1...247) 如果是广播，则为 0	16 （请求代码）
	3	要写入的第一个字的地址	
	4	N = 要写入的字数	
	5	00 （未发送的字节，偏移影响）	2*N = 要写入的字节数
	6	要写入第一个字的值	
	7	要写入的第二个值	
	...	...	
	N+5	要写入的 N 个值	
接收表 （响应后）	N+6	从站 @(1...247)	16 （响应代码）
	N+7	已写入的第一个字的地址	
	N+8	已写入的位数 (= N)	

注意： 传输偏移 = 7 表示抑制已发送帧中的第七个字节。这还可以使传输表中字的值进行良好的通讯。

Modbus 请求：读取设备标识

下表表示请求 43 （读取设备标识）：

梯级	指令	注释
0	LD 1 [%MW800:=16#0106] [%MW801:=16#0000] [%MW802:=16#032B] [%MW803:=16#0E01] [%MW804:=16#0000]	[%MW800:=16#0106]：标准 Modbus 标头 [%MW801:=16#0000]：无传输和接收偏移 [%MW802:=16#032B]：从站地址，功能代码 [%MW803:=16#0E01]：MEI 类型，读取设备 ID 代码 [%MW804:=16#0000]：对象 ID，未使用

Modbus 请求：诊断

下表表示请求 8（诊断）：

梯级	指令	注释
0	LD 1 [%MW1000:=16#0106] [%MW1001:=16#0000] [%MW1002:=16#0308] [%MW1003:=16#0000] [%MW1004:=16#1234]	[%MW1000:=16#0106]：标准 Modbus 标头 [%MW1001:=16#0000]：无传输和接收偏移 [%MW1002:=16#0308]：从站地址，功能代码 [%MW1003:=16#0000]：子功能代码 [%MW1004:=16#1234]：任何数据 从站应答将是请求副本。此模式称为回显或镜像模式。

示例 1：编写 Modbus 应用程序

主站程序：

梯级	指令	注释
0	LD 1 [%MW0:=16#0106] [%MW1:=16#0300] [%MW2:=16#0203] [%MW3:=16#0000] [%MW4:=16#0004]	[%MW0:=16#0106]：传输长度 = 6 [%MW1:=16#0300]：接收偏移 = 3，传输偏移 = 0 %MW2 至 %MW4：传输 [%MW2:=16#0203]：从站 2，功能 3（读取多个字） [%MW3:=16#0000]：在从站中要读取的第一个字的地址：至 0 地址 [%MW4:=16#0004]：要读取的字数：4 个字 （%MW0 至 %MW3）
1	LD 1 AND %MSG2.D [EXCH2 %MW0:11]	–
2	LD %MSG2.E ST %Q0.0 END	–

从站程序：

梯级	指令	注释
0	LD 1 [%MW0:=16#6566] [%MW1:=16#6768] [%MW2:=16#6970] [%MW3:=16#7172] END	–

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

使用 SoMachine Basic，可同时为主站和从站创建应用程序。对于从站，仅将一些存储器字写入一组已知值。在主站中，将**交换**指令的字表初始化为从 Modbus 地址 2 处的从站读取从位置 %MW0 开始的 4 个字。

注意：请注意，应使用 Modbus 主站应用程序 %MW1 中的接收偏移设置。偏移为 3 将在接收表中的第三个位置添加一个字节 （值 = 0）。这将对齐主站中的字，以使它们处于正确的字边界。如果没有此偏移，则将会在**交换**块中拆分两个字之间的每个数据字。使用此偏移的目的是使操作更方便。

在执行 **EXCH2** 指令之前，应用程序将检查与 %MSG2 关联的通讯位。最后，将在本地基板控制器 I/O 的第一个输出位上感测和存储 %MSG2 的错误状态。也可以添加使用 %SW64 进行的其他错误检查过程，以使其更精确。

动态数据表以在线模式初始化对应的接收表部分：

地址	值	格式
%MW5	0203	十六进制
%MW6	0008	十六进制
%MW7	6566	十六进制
%MW8	6768	十六进制
%MW9	6970	十六进制
%MW10	7172	十六进制

在下载并设置每个控制器使其运行后，打开主站上的动态数据表。检查表的响应部分，以确定响应代码为 3 以及已读取正确的字节数。同样，在本示例中，从从站读取的字 （%MW7 开始）应与主站中的字边界正确对齐。

示例 2：编写 Modbus 应用程序

主站程序：

梯级	指令	注释
0	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]	[%MW0:=16#010C]：传输表长度：0C （十六进制） = 12 （十进制），从 %MW2 到 %MW7 [%MW1:=16#0007] [%MW2:=16#0210]：从站地址 2， 10h 功能代码写入字 [%MW3:=16#0010]：从站中的地址 16 [%MW4:=16#0002]：写入 2 个字 [%MW5:=16#0004]：要写入的字节数 [%MW6:=16#6566]：第一个字的值 [%MW7:=16#6768]：第二个字的值
1	LD 1 AND %MSG2.D [EXCH2 %MW0:12]	–
2	LD %MSG2.E ST %Q0.0 END	–

从站程序：

梯级	指令	注释
0	LD 1 [%MW18:=16#FFFF] END	–

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

使用 SoMachine Basic，可同时为主站和从站创建应用程序。对于从站，写入单个存储器字 %MW18。这将在从站上为 %MW0 至 %MW18 的存储器地址分配空间。如果未分配空间， Modbus 请求将尝试写入从站上不存在的位置。

在主站中，将 EXCH2 指令的字表初始化为向 Modbus 地址 2 处的从站的地址 %MW16 （十六进制值 10）读入 4 个字节。

注意：请注意，应使用 Modbus 主站应用程序 %MW1 中的传输偏移设置。偏移为 7 将抑制第 6 个字的高字节 （%MW5 中的十六进制值 00）。此举实现在字表的传输表中对齐数据值，以使它们处于正确的字边界。

在执行 EXCH2 指令之前，应用程序将检查与 %MSG2 关联的通讯位。最后，将在本地基板控制器 I/O 的第一个输出位上感测和存储 %MSG2 的错误状态。也可以添加使用 %SW64 进行的其他错误检查过程，以使其更精确。

主站上的动态数据表初始化：

地址	值	格式
%MW0	010C	十六进制
%MW1	0007	十六进制
%MW2	0210	十六进制
%MW3	0010	十六进制
%MW4	0002	十六进制
%MW5	0004	十六进制
%MW6	6566	十六进制
%MW7	6768	十六进制
%MW8	0210	十六进制
%MW9	0010	十六进制
%MW10	0004	十六进制

从站上的动态数据表初始化：

地址	值	格式
%MW16	6566	十六进制
%MW17	6768	十六进制

下载并设置每个控制器使其运行后，打开从站控制器上的动态数据表。将 %MW16 和 %MW17 中的两个值写入从站。

在主站中，动态数据表可以用于检查交换数据的接收表部分。该数据显示了从站地址、响应代码、写入的第一个字和以上示例中在 %MW8 处开始写入的字数。

章 5
计划块 (%SCH)

使用计划块

本节介绍如何使用 Schedule blocks 并提供其编程指南。

本章包含了哪些内容?

本章包含了以下主题：

描述

简介

Schedule blocks 用于在预定义的月份、日期和时间控制动作。

Schedule blocks 仅在 SoMachine Basic 中配置，且不会以与其他功能块相同的方式插入到程序梯级中。

注意： 检查系统位 %S51 和系统字 %SW118，以确认实时时钟 (RTC) 选件已安装。使用 Schedule blocks 要求安装 RTC 选件。

参数

要配置参数，请按照配置功能块过程 (参见第 *129* 页) 进行操作，并阅读内存分配模式中的说明。

Schedule blocks 具有以下参数：

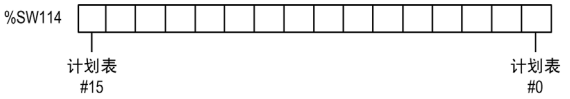
参数	描述	值
已使用	已使用的地址	如果选择此参数，则当前在程序中使用此地址。
地址	Schedule blocks 对象地址	程序只能包含数量有限的 Schedule blocks 对象。有关 Schedule blocks 的最大数量，请参阅硬件平台的编程指南。
已配置	是否已配置所选 Schedule blocks 数以供使用。	若选中该复选框，则对其进行配置以供使用。否则，则未使用。
输出位	输出位	输出赋值由以下 Schedule blocks 激活：%Mi 或 %Qj.k。 当前日期和时间介于活动周期的起始设置和结束设置之间时，该输出将设为 1。
开始日期	月份中启动 Schedule blocks 的日期。	**1...31**
开始月	启动 Schedule blocks 的月份。	Schedule blocks。
结束日期	月份中结束 Schedule blocks 的日期。	**1...31**
结束月	结束 Schedule blocks 的月份。	**一月 ... 十二月**
开始时间	启动 Schedule blocks 的当天的时间，小时和分钟。	小时：**0...23** 分钟：**0...59**
结束时间	结束 Schedule blocks 的当天的时间，小时和分钟。	小时：**0...23** 分钟：**0...59**

参数	描述	值
星期一	复选框，用于确定激活 Schedule blocks 的星期几。	若选中该复选框，则对其进行配置以供使用。 否则，则未使用。
星期二		
星期三		
星期四		
星期五		
星期六		
星期日		
注释	注释	可以将注释与此对象进行关联。

启用计划块

系统字 %SW114 的位将启用 （将位设置为 1）或禁用 （将位设置为 0） 16 个 Schedule blocks 中每一个块的操作。

%SW114 中的 Schedule blocks 分配：



默认情况下 （或冷重启之后），该系统字的所有位都将设为 1。用户可通过程序选择性地使用这些位。

计划块的输出

如果同一输出 （%Mi 或 %Qj.k）由多个块分配，则最终分配到此对象的是每个块的结果的 OR 操作 （同一输出可能有多个 Schedule blocks）。

例如，计划块 %SCH0 和 %SCH1 都已被分配到输出 %Q0.0。 %SCH0 将输出设置为从星期一的 12:00 到 13:00， %SCH1 将输出设置为从星期二的 12:00 到 13:00。结果是将输出同时设置为星期一和星期二的 12:00 到 13:00。

编程和配置

简介

Schedule blocks 用于在预定义的月份、日期和时间控制动作。

编程示例

下表显示了夏季月份喷洒程序示例的参数：

参数	值	描述
地址	**实时时钟 6**	Schedule blocks 编号 6
已配置	选中框	选中框可配置 Schedule blocks 编号 6。
输出位	**%Q0.2**	激活输出 %Q0.2
开始日期	**21**	在六月份的第 21 天启动活动
开始月	**六月**	在六月份启动活动
开始时间	**21**	在 21:00 启动活动
结束日期	**21**	在九月的第 21 天停止活动
结束月	**九月**	在九月结束活动
结束时间	**22**	在 22:00 停止活动
星期一	选中框	在星期一运行活动
星期二	未选中框	无活动
星期三	选中框	在星期三运行活动
星期四	未选中框	无活动
星期五	选中框	在星期五运行活动
星期六	未选中框	无活动
星期日	未选中框	无活动

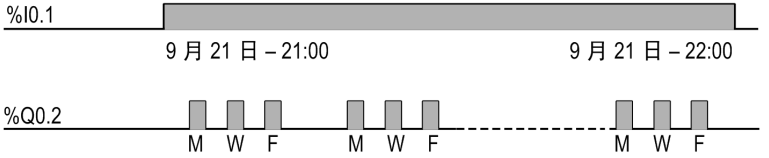
使用此程序，可通过开关或连接至输入 %I0.1 的湿度计禁用 Schedule blocks：

梯级	指令	注释
0	LD %I0.1 ST %SW114:X6	在本例中， %SCH6 已验证。

注意： 请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

时序图

下列时序图显示了如何激活输出 %Q0.2：



章 6
PID 功能

本章包含了哪些内容？

本章包含了以下部分：

节 6.1
PID 操作模式

PID 操作模式

简介

SoMachine Basic PID 控制器提供四种不同的操作模式，用户可在 SoMachine Basic 中 **PID 助手**的**常规**选项卡 (参见第 *223* 页) 上对其进行配置。

PID 操作模式有：

- PID 模式
- AT + PID 模式
- AT 模式
- 字地址

PID 模式

当 PID 控制器启动时，缺省情况下简单 PID 控制器模式处于活动状态。要成功控制此流程，须提前获取将在 **PID** 选项卡 (参见第 *227* 页) 上指定的增益值 Kp、Ti 和 Td。您可以在 **PID 助手**屏幕 (参见第 *221* 页) 的 **PID** 选项卡上选择校正器类型的控制器 （PID 或 PI）。若选择 PI 校正器类型，则会禁用微分时间 **Td** 字段。

若使用 PID 模式，会禁用自调节功能且无法使用**助手配置**屏幕上的 **AT** 选项卡 (参见第 *228* 页)。

AT + PID 模式

在此模式下，当 PID 控制器启动时，自调节功能将处于活动状态。随后，自调节功能将计算增益值 Kp、 Ti 和 Td (参见第 *227* 页) 以及 PID 操作 (参见第 *230* 页) 的类型。在自调节序列的末尾，控制器将使用自调节计算所得的参数切换到已调整设置点的 PID 模式。

如果自动调节算法检测到错误 (参见第 *236* 页)：

- 不对任何 PID 参数进行计算。
- 自调节输出将设置为启动自调节前应用到流程的输出。
- **PID 状态列表**下拉列表中显示错误消息。
- 取消 PID 控制。

当处于 AT + PID 模式时，从自调节转换为 PID 模式是自动无缝进行的。

AT 模式

在此模式下，自调节功能将在 PID 控制器启动时激活，并自动计算增益值 Kp、 Ti 和 Td (参见第 *227* 页) 以及 PID 操作 (参见第 *230* 页) 的类型。在完成自调节流程并成功确定 Kp、 Ti 和 Td 参数以及 PID 操作 (参见第 *230* 页) 的类型后 （或在自调节算法中检测到错误后），自调节数字输出将设置为 0，并在 PID 状态列表 (参见第 *236* 页) 下拉列表中显示**自调节完成**的消息。然后，PID 控制器停止并等待。计算得出的 Kp、Ti 和 Td PID 系数可在其各自的存储器字 (%MWx) 中使用。

字地址

该 PID 模式是通过将所需值赋给与该选择关联的字地址实现的：

- %MWxx = 0：控制器已禁用。
- %MWxx = 1：控制器在简单 PID 模式下运行。
- %MWxx = 2：控制器在 AT+ PID 模式下运行。
- %MWxx = 3：控制器仅在 AT 模式中运行。
- %MWxx = 4：控制器运在简单 PID 模式下运行，带有 PI 修正器类型。

此 word address 模式让您能够通过应用程序管理 PID 控制器操作模式，从而能够适应您的要求。

节 6.2
PID 自调节配置

PID 自调节配置

简介

本节逐步介绍使用自调节 (AT) 配置 SoMachine Basic PID 控制器所需的所有步骤。

本节包含以下步骤：

步骤	主题
1	配置模拟量通道 (参见第 *206* 页)
2	PID 配置的先决条件 (参见第 *206* 页)
3	配置 PID (参见第 *207* 页)
4	控制设置 (参见第 *208* 页)

步骤 1：配置模拟量通道

PID 控制器使用模拟量反馈信号 （称为过程值）来计算用于控制过程的算法。 Logic Controller 具有可用于获取该过程值的嵌入式模拟量输入。有关模拟量输入配置的更多详细信息，请参阅 Logic Controller 的*编程指南*。

如果模拟量输出正用于驱动要控制的系统，请确保正确配置该模拟量输出。请参阅 Logic Controller 的模拟量输出扩展模块。

步骤 2：PID 配置的先决条件

在配置 PID 控制器前，请确保已执行以下阶段：

相位	描述
1	已在程序中启用 PID (参见第 *233* 页)。
2	扫描模式已设置为定期 (参见第 *235* 页)。

步骤 3：配置 PID

将固态输出与 PID 功能结合使用。使用输出继电器可能导致快速超过其使用寿命周期极限，导致触点冻开或者焊接关闭，进而造成继电器无法运行。

⚠ 警告

设备无法操作或意外的设备操作

- 请勿将继电器输出与 PID 功能搭配使用。
- 如果需要数字量输出以驱动要控制的系统，则仅使用固态输出。

不遵循上述说明可能导致人员伤亡或设备损坏。

要用自调节实现 PID 控制器，请执行以下步骤：

步骤	操作
1	在 **PID 助手**屏幕 （离线模式）的 **General** tab (参见第 *223* 页) 中，选择 **AT+PID** （或 **AT**），或从 Operating Modes (参见第 *204* 页) 中选择可将关联字设置为 2 或 3 的**字地址**。
2	选中与 **PID 状态**关联的框，并在字段中输入存储器字的地址。
3	在 **Input** tab (参见第 *226* 页) 中，输入用作测量值的模拟量输入的地址。
4	如果需要**转换**或**报警**，请参阅 **PID 助手**屏幕的**输入**选项卡 (参见第 *226* 页)。
5	在 **PID** 选项卡 (参见第 *227* 页) 中，输入设置点的值。通常，该值为存储器地址或模拟量输入。
6	**PID** 选项卡中的**修正器类型**必须设置为 **PID** 或 **PI**。
7	设置 **PID** 选项卡中的**参数**：**Kp (x0,01)**、**Ti (x0,1s)** 和 **Td (x0,1s)**。如果 **AT+PID** 或 **AT** 处于运行模式 (参见第 *204* 页)，这些参数应为**存储器字地址 (%MWxx)**，以便自调节算法填入这些参数的计算值。
8	在 **PID** 选项卡中输入 PID **采样周期** (Ts (参见第 *214* 页))。**采样周期**是关键参数，必须仔细确定。
9	在 **AT** 选项卡中，**AT 模式**在缺省情况下必须设置为**授权**。从列表中选择**动态修正器**，该列表包含**快速**、**中速**、**慢速**或**字地址**修正器类型。有关详细信息，请参阅 PID 助手 (参见第 *221* 页) 中的 **AT** 选项卡。
10	在 **AT** 选项卡中，输入 **AT 触发器**存储器位，以存储自调节期间步进变化的值。有关详细信息，请参阅 PID 助手 (参见第 *221* 页) 中的 **AT** 选项卡。
11	在 **Output** tab (参见第 *230* 页) 中，将**动作**设置为列表中的**位地址**。在**位**字段中输入存储器位地址。如有需要，**限制**可在 “ 输出 ” 选项卡 (参见第 *230* 页) 中进行配置。在**模拟量输出**字段中，设置字地址：模拟量输出或存储器字。将 **Output PWM** (参见第 *230* 页) 设置为**授权**。在手动模式中，在**周期 （0.1 秒）**字段中输入值，或在**输出**字段中输入输出的存储器字地址。有关手动模式操作的详细信息，请参阅 “ 输出 ” 选项卡 (参见第 *230* 页)。
12	单击**确定**以确认 PID 控制器配置。

步骤 4：控制设置

将固态输出与 PID 功能结合使用。使用输出继电器可能导致快速超过其使用寿命周期极限，导致触点冻开或者焊接关闭，进而造成继电器无法运行。

⚠ 警告

设备无法操作或意外的设备操作

- 请勿将继电器输出与 PID 功能搭配使用。
- 如果需要数字量输出以驱动要控制的系统，则仅使用固态输出。

不遵循上述说明可能导致人员伤亡或设备损坏。

要以 AT+PID 操作模式 (参见第 *204* 页) 开始操作，请执行以下步骤：

步骤	操作
1	将 PC 连接到控制器并传输应用程序。
2	将控制器切换到 “ 运行 ” 模式。

注意：在将控制器切换到 “ 运行 ” 模式之前，请检查机器的操作情况是否允许为应用程序的其余部分切换到 “ 运行 ” 模式。

步骤	操作
1	创建动态数据表，该表包含配置期间所定义的对象。有关动态数据表创建的详细信息，请参阅 *SoMachine Basic 操作指南*。

2	验证过程值和应用程序值之间的一致性。该测试非常重要，因为 PID 控制器的操作是否成功取决于测量值的准确性。如果对测量值的准确性存在任何疑问，请将 Logic Controller 设置为 “ 停止 ” 状态并验证模拟量通道的接线。
	如果执行器不受控制：
	● 对于模拟量输出，请验证模拟量通道的输出电压或电流。
	● 对于 PWM 输出，请验证：
	  ● 专用输出的 LED 是否亮起
	  ● 电源和 0 伏电路的接线
	  ● 执行器电源是否正在供电

3	在动态数据表中，请验证： ● 输出模式是否已设置为自动。 ● 应用程序所需的所有参数是否都已设置为适当的值。
4	设置 Logic Controller 扫描周期，以使 PID 控制器的**采样周期** (Ts) 值为扫描周期的整数倍。有关如何确定采样周期的详细信息，请参阅调节 PID (参见第 *214* 页)。

步骤	操作
5	自调节序列完成后，参数 **Kp**、**Ti** 和 **Td** 将写入 Logic Controller 的 RAM 存储器。只要应用程序有效（电源关闭不满 30 天）且未执行冷启动，这些值将一直保存。

每当在 **AT 触发器**存储器位检测到上升沿，自调节过程就会重复。

节 6.3
PID 标准配置

本节包含了哪些内容？

本节包含了以下主题：

PID 字地址配置

简介

本节将指导您完成使用字地址操作模式 (参见第 *204* 页) 配置 SoMachine Basic PID 控制器所需的所有步骤。该模式的使用灵活性要优于其他 PID 模式。

本节包含以下步骤：

步骤	主题
1	PID 配置的先决条件 (参见第 *211* 页)
2	配置 PID (参见第 *211* 页)
3	控制设置 (参见第 *212* 页)

步骤 1：PID 配置的先决条件

在配置 PID 前，请已执行确保以下阶段：

相位	描述
1	已配置模拟量输入和模拟量输出 （如有需要）。请参阅 Logic Controller 的*编程指南*。
2	已在程序中启用 PID (参见第 *233* 页)。
3	扫描模式已设置为定期 (参见第 *235* 页)。

步骤 2：配置 PID

将固态输出与 PID 功能结合使用。使用输出继电器可能导致快速超过其使用寿命周期极限，导致触点冻开或者焊接关闭，进而造成继电器无法运行。

⚠ 警告

设备无法操作或意外的设备操作

- 请勿将继电器输出与 PID 功能搭配使用。
- 如果需要数字量输出以驱动要控制的系统，则仅使用固态输出。

不遵循上述说明可能导致人员伤亡或设备损坏。

以下步骤解释了如何在**字地址**模式下实现 PID 控制器。有关如何配置 PID 的详细信息，请参阅 PID 助手 (参见第 *221* 页) 一节。

对于 PID 参数的动态修改 （离线和在线模式下），请在关联字段中输入存储器地址，这样就无需切换至离线模式进行值的实时更改。

步骤	操作
1	在 **PID Assistant** 屏幕的**常规**选项卡中 （离线模式下），选择 **Operating Modes** 下拉列表中的**字地址**。选中与 PID 状态关联的框，并在字段中输入存储器字的地址。
2	在**输入** (参见第 *226* 页) 选项卡中，输入用作测量值的模拟量输入的地址。如果需要**转换**或**报警**，请参阅 PID 助手 (参见第 *221* 页) 的**输入**选项卡 (参见第 *226* 页)。
3	在 **PID** 选项卡中，输入**设置点**的值。通常，该值为存储器地址或模拟量输入。**参数** （Kp、 Ti 和 Td）应为**存储器字地址 (%MWxx)**。 在 PID 选项卡 (参见第 *227* 页) 中输入 PID **采样周期** (Ts (参见第 *227* 页))。该参数也可以是存储器字 （然后，值可以使用动态数据表来设置）。 在**字地址**操作模式中，**修正器类型**被设置为**自动**且呈灰色 （不能对其进行手动修改）。
4	在 **AT** 选项卡中，应选中 AT 模式，以便进行**授权**。输入**动态修正器**和 **AT 触发器**。有关详细信息，请参阅 **PID 助手**屏幕中的 **AT** 选项卡 (参见第 *228* 页)。
5	在**输出**选项卡中，**操作**应设置为**位地址**。输入**存储器位地址**。如有需要，可在 “ 输出 ” 选项卡 (参见第 *230* 页) 中配置**限制**。在**模拟量输出**字段中，设置字地址：模拟量输出或存储器字。如有需要，请设置**输出 PWM**，请参阅 PID 助手 (参见第 *221* 页) 中的**输出**选项卡 (参见第 *230* 页)。
6	单击**确定**以确认 PID 控制器配置。

步骤 3：验证设置

步骤	操作
1	将 PC 连接到 Logic Controller 并传输应用程序。
2	将 Logic Controller 切换到 “ 运行 ” 模式。

注意：在将逻辑控制器切换到 “ 运行 ” 模式之前，请验证机器的操作情况是否允许为应用程序的其余部分切换到 “ 运行 ” 模式。该过程与 AT 和 AT+PID 操作模式下使用的过程相同。字地址配置使您可以通过软件来修改 PID 操作模式。对于 PID 模式，该过程非常简单，如果参数 （Kp、 Ti、 Td 和 Ts）已知，则无需执行自调节。

下表提供了设置 PID 控制器的一般步骤

步骤	操作
1	创建动态数据表，该表包含配置期间所定义的对象。有关详细信息，请参阅 *SoMachine Basic 操作指南*。

2	验证过程值和动态数据表所定义的其他值是否一致。如果在测量值的准确度方面存在任何疑问，请将逻辑控制器设置为“停止”并验证模拟量通道的接线。
	如果发现执行器不受控制，请执行以下操作：
	● 对于模拟量输出，请验证模拟量通道的输出电压或电流。
	● 对于 PWM 输出，请验证：
	  ● 专用输出的 LED 是否亮起
	  ● 电源和 0 伏电路的接线是否正确
	  ● 执行器电源是否正在供电

3	设置 Logic Controller 扫描周期，以使 PID 控制器的**采样周期** (Ts) 为扫描周期的整数倍。有关采样周期的详细信息，请参阅确定采样周期 (参见第 *219* 页)
4	如果计划使用自调节 (参见第 *214* 页) 功能，您可能需要运行手动模式 (参见第 *217* 页) 以确定 **PID 助手**的 **AT** 选项卡 (参见第 *228* 页) 所定义的**动态修正器**和 **AT 触发器**。
5	使用动态数据表为回路控制器加电： ● 设置操作模式 (参见第 *204* 页)。 ● 启用 PID 控制器 (参见第 *235* 页)。 ● 根据所选操作模式，将配置 (参见第 *211* 页) 阶段所定义的值设置为合适的值。

使用自调节 (AT) 进行 PID 调节

简介

自调节模式允许 Kp、 Ti、 Td 和操作参数进行自动调节，以实现 PID 功能的优化整合。SoMachine Basic 提供的自调节功能尤其适合热过程的自动调节。

本节包含以下主题：

- 自调节要求
- 自调节过程的描述
- 存储计算所得的系数
- 调整 PID 参数
- 自调节的重复性
- 自调节和 PID 控制使用限制

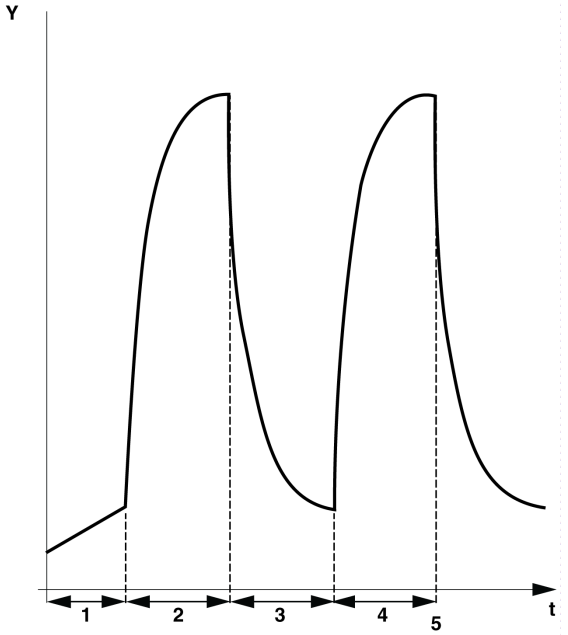
自调节要求

使用自调节功能时，请确保控制过程和 Logic Controller 满足以下要求：

- 过程要求：
  - 过程必须是一个稳定的开放回路系统。
  - 过程在整个操作范围内必须大部分都是线性的。
  - 对模拟量输出进行级别更改的过程响应遵循瞬变渐近模式。
  - 过程处于稳定状态，且在自调节序列开始时的输入为空。
  - 整个过程都必须没有干扰。否则，计算得出的参数将是错误的，也可能会导致自调节过程无法正确运行。
- 配置要求：
  - 将 Logic Controller 配置为定期扫描模式，以确保自调节功能的正确运行。
  - 仅在没有其他 PID 控制器运行时使用自调节功能。
  - 将 Kp、 Ti 和 Td 系数配置为存储器字地址 (%MWxx)。
  - 将**输出**选项卡中的动作类型设置为存储器位地址 (%Mxx)。

自调节过程的描述

自调节过程分为 4 个连续的阶段。要成功完成自调节，必须执行该过程的所有阶段。下面的过程响应曲线和表描述了 SoMachine Basic PID 自调节功能的 4 个阶段：



下表描述了自调节阶段：

自调节阶段	描述
1	稳定性阶段 (1) 在自调节过程启动之后开始。在该阶段中，自调节功能执行检查以确保过程值处于稳定状态。
2	第一个步进变化应用于过程。该阶段将生成与上图显示的响应类似的过程步进响应。
3	松弛阶段 (1) 在第一个步进响应稳定之后开始。
4	第二个步进变化将以上述阶段 2 相同的数量和方式应用于程序。自调节过程结束。过程值将恢复 (1)，自调节参数也将计算得出并存储在各自的存储器字中。请参阅 “ 存储计算所得的系数 ” 描述 (参见第 *216* 页)。
(1) 自调节开始之前最后应用于过程的输出被用作自调节过程的开始点和松弛点。	

存储计算所得的系数

自调节序列完成后，将使用计算所得的值设置分配给 Kp、 Ti 和 Td 系数的存储器字及动作类型。这些值将写入 RAM 存储器，只要应用程序有效且不执行冷启动 (%S0)，这些值将一直保存在 Logic Controller 中。

如果系统未受到外部干扰，计算所得的值可能会写入 PID 控制器的设置中 （请参阅 PID 助手 (参见第 *230* 页) 的 **PID** 选项卡）。这样， PID 控制器操作模式可以设置为 PID 模式。

调整 PID 参数

自调节方法可能会提供一个非常动态的命令，从而导致设置点步进变化期间发生有害过冲。要优化从自调节获取的 PID 参数 （Kp、 Ti、 Td）所提供的过程调节，您还可以直接从 **PID 助手**屏幕的 **PID** 选项卡中或通过相应的存储器字 (%MW) 对这些参数值进行手动调整。有关手动参数调整的更多详细信息，请参阅附录 (参见第 *247* 页)。

自调节的重复性

在 **AT** 选项卡中， **AT 触发器**可以实现自调节序列的重复性。自调节过程将在已链接至 **AT 触发器**的信号的各个上升沿上启动。

自调节使用限制

热过程常被吸收到一阶纯延迟模型中。此类模型可用两个关键参数来描述：

- 时间常量 τ
- 延迟时间 θ

自调节最适合时间常量 (τ) 和延迟时间 (θ) 满足以下条件的过程：

- 10 秒 < (τ + θ) < 2700 秒 （例如， 45 分钟）
- 2 < τ / θ < 20

手动模式

简介

可通过 **PID 助手**屏幕 (**Output** tab (参见第 *230* 页)) 访问手动模式。此模式可用于从 PID 绕过顺序。使用手动模式可以实现两个主要目标：

- 初始化设置
- 确定采样周期。

描述

手动模式可以让您指定 **Output** value (参见第 *230* 页)。此操作特别适合用于测试系统响应能力。

在**输出** (参见第 *230* 页) 选项卡上，将**位地址**设置为 1 以激活手动模式。如果设置为 “ 允许 ”，则手动模式是唯一可访问的模式。

应用程序

当手动模式处于活动状态时，输出将分配有您设置的固定值。该输出值介于 0 至 10,000 之间 （对于 PWM 输出，该值为 0 至 100%）。

此外，您也可以使用手动模式进行试用，以确定最小 / 最大输出限制。

手动模式也需要使用过程曲线响应方法 (参见第 *219* 页)，以帮助找到正确的采样时间 (Ts)。

启动手动模式

在启动手动模式之前，请务必确保 Logic Controller“ 运行 / 停止 ” 开关处于 “ 运行 ” 位置。

使用动态数据表启动手动模式：

步骤	描述
1	通过将专用存储器位设置为 1 启用手动模式。有关详细信息，请参阅**输出**选项卡 (参见第 *230* 页)。
2	如果使用 PWM，请将 PWM 期间设置为所需的值。
3	在 **PID 助手**屏幕的**常规**选项卡 (参见第 *223* 页) 上，将与操作模式关联的存储器字设置为 1 （PID 模式）。有关使用字地址的操作模式的详细信息，请参阅操作模式说明 (参见第 *204* 页)。
4	在**输出**选项卡 (参见第 *230* 页) 上，将与手动输出关联的存储器字设置为所需的值。在系统处于其初始状态的条件下，可以多次选择该手动设置点值。
5	启用回路控制器 (参见第 *211* 页)。

停止手动模式

使用动态数据表停止手动模式：

步骤	描述
1	禁用回路控制器 (参见第 *211* 页)。
2	通过将专用存储器位设置为 0 禁用手动模式。有关详细信息，请参阅**输出**选项卡 (参见第 *230* 页)。
3	在 PID 控制器的**常规**选项卡 (参见第 *223* 页) 上设置与操作模式关联的存储器字。有关使用字地址的操作模式的详细信息，请参阅操作模式说明 (参见第 *204* 页)。
4	在**输出**选项卡 (参见第 *230* 页) 上，将与手动输出关联的存储器字设置为 0。

确定采样周期 (Ts)

简介

采样周期 (Ts) 是 PID 调节的关键参数。应在 **PID 助手**屏幕的 **PID** 选项卡 (参见第 *227* 页) 中小心地设置采样周期 (Ts)。该参数与要控制的过程的时间常量 (τ) 密切相关。

本节将介绍如何使用在线模式，还将介绍采用周期 (Ts) 的两种确定方式：

- 过程响应曲线方式，
- 反复试验方式。

过程响应曲线方式

这种方式是一个开放回路过程，旨在确定要控制的过程的时间常量。首先，需要确保过程可以通过一阶时间延迟模型来描述。其原理非常简单：在记录过程输出曲线时，将步进变化应用于过程的输入。然后，使用图形方式确定过程的时间延迟。

要使用过程响应曲线方式来确定采样周期 (Ts)，请执行以下操作：

步骤	操作
1	这一点是以您已配置 PID 的**常规**、**输入**、 **PID**、 **AT** 和 **输出**选项卡中的各个设置为前提。
2	选择 **PID 助手**屏幕中的**输出**选项卡 (参见第 *230* 页)。
3	选择**手动模式**下拉列表中的**允许**或**地址位**，以对手动输出进行授权。
4	将 “ 输出 ” 字段设置为高级别 （在 [5,000...10,000] 范围内）。
5	将应用程序下载到 Logic Controller。有关如何下载应用程序的详细信息，请参阅 *SoMachine Basic 操作指南*。
6	运行 PID 并检查响应曲线是否上升。
7	响应曲线达到稳定状态时，停止 PID 测量。
8	使用以下图形方式来确定控制过程的时间常量 (T)： 1. 使用以下公式计算上升到 63% 时 ($S_{[63\%]}$) 的过程值输出：$S_{[63\%]} = S_{[初始]} + (S_{[结束]}-S_{[初始]})x63\%$ 2. 利用图形计算对应于 S(63%) 的时间横坐标 ($t_{[63\%]}$)。 3. 利用图形计算对应于过程响应上升起始位置的初始时间 ($t_{[初始]}$)。 4. 使用以下关系计算控制过程的时间常量 (τ)：$\tau = t_{[63\%]}-t_{[初始]}$
9	根据上一步所确定的 (τ) 值，使用以下规则计算采样周期 (Ts)(1)：$Ts = \tau/75$
10	设置定期扫描模式的扫描周期，以使采样周期 (Ts) 为扫描周期的整数倍：扫描周期 = Ts / n，其中 n 为正整数 (2)
(1) 采样周期的基本单位为 10 毫秒。因此，您应对 Ts 的值进行舍入操作，以使其最接近于 10 毫秒。 **(2)** 您必须选择 “n”，以使得出的扫描周期为位于 [2...150] 毫秒范围内的正整数。	

反复试验方式

反复试验方式需要向自调节功能提供连续的采样周期猜测值，直到算法成功整合出令人满意的 Kp、 Ti 和 Td 值。

注意：与过程响应曲线方式不同，反复试验方式并非基于过程响应的任何近似法则。但是，它的优点在于：它能整合出与实际值位于同一数量级的采样周期值。

要执行自调节的反复试验估算，请执行以下操作：

步骤	操作
1	选择 PID 配置窗口中的 **AT** 选项卡。
2	将自调节的**输出限制**设置为 **10,000**
3	将应用程序下载到 Logic Controller。有关如何下载应用程序的详细信息，请参阅 SoMachine Basic 操作指南 *(参见 SoMachine Basic, Operating Guide)*。
4	选择 **PID 助手**屏幕中的 **PID** 选项卡。
5	在**采样周期** (1) 字段中提供第一个或第 n 个猜测值。
6	启动自调节 (参见第 *206* 页)。
7	稍侯，直到自调节过程结束。
8	可能会出现两种情况： ● **自调节成功完成：**继续执行步骤 10。 ● **自调节失败：**请参阅自调节检测到的错误代码 (参见第 *237* 页)。这意味着采样周期 (Ts) 的当前猜测值不正确。根据需要，多次尝试新的 Ts 猜测值并重复步骤 3 至步骤 8，直到自调节过程最终整合。
9	请根据以下原则提供新的 Ts 猜测值： ● 自调节结束，其检测到的错误代码为 800C （十六进制）。这意味着采样周期 Ts 太大。减小 Ts 值，以便提供新的猜测值。 ● 自调节结束，其检测到的错误代码为 800A （十六进制）。这意味着采样周期 Ts 太小。增大 Ts 值，以便提供新的猜测值。
10	根据需要在 **PID 助手**屏幕的 PID 选项卡 (参见第 *227* 页) 中调整 PID 控制参数 (2) （Kp、 Ti 和 Td）。

(1) 如果您未掌握采样周期可能范围的初步迹象，请将此值设置为可能的最小值：1 （10 毫秒为 1 个单位）。
(2) 如果该组控制参数提供的 PID 调节不能提供完全令人满意的结果，您可以不断进行采样周期反复试验求值优化，直到得出一组正确的 Kp、 Ti 和 Td 控制参数。

在线模式

处于在线模式时，如果 Logic Controller 有定期任务， Ts 字段中所显示的值 （在 **PID 助手**屏幕 (参见第 *221* 页) 中）可能不同于所输入的参数 (%MW)。 Ts 值是定期任务的整数倍，而 %MW 值是 Logic Controller 所读取的值。

节 6.4
PID 助手

本节包含了哪些内容?

本节包含了以下主题:

访问 PID 助手

简介

通过使用 SoMachine Basic 的 **PID 助手**窗口，用户可配置 PID 控制器。

配置助手

在 PID 属性表中，单击**配置 [...]** 按钮。 **PID 助手**屏幕随即出现。

PID 助手屏幕如下图所示：

根据您是处于离线或在线模式， **PID 助手**屏幕可显示多个不同的选项卡。

选项卡	访问模式	链接
常规	离线	“ 常规 ” 选项卡 (参见第 *223* 页)
输入	离线	“ 输入 ” 选项卡 (参见第 *226* 页)
PID	离线	“PID” 选项卡 (参见第 *227* 页)
AT	离线	“AT” 选项卡 (参见第 *228* 页)
输出	离线	“ 输出 ” 选项卡 (参见第 *230* 页)

选择某一操作模式后，包含空字段的选项卡便会如  中所示进行显示，且字段的边缘将填充为红色。

常规”选项卡

简介

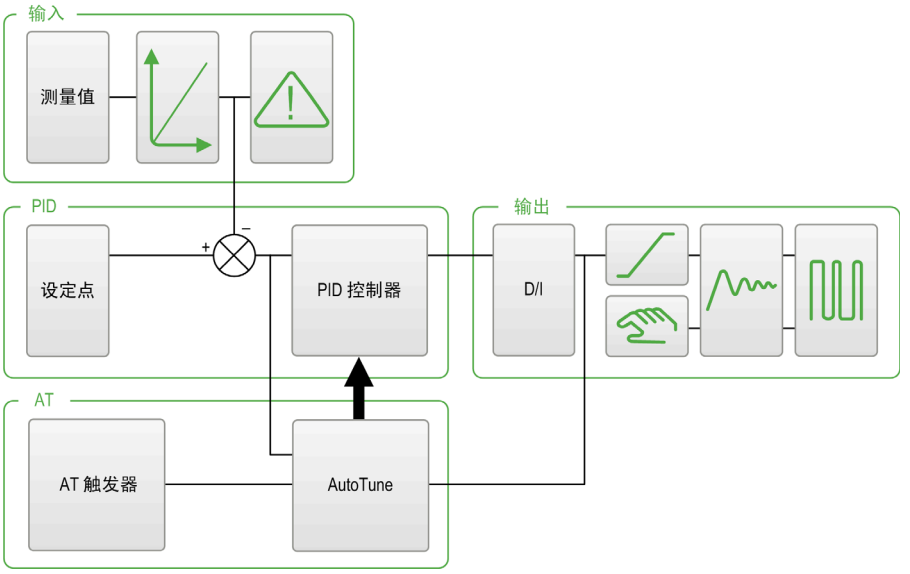
本节介绍 PID 的**常规**选项卡。当您以离线模式访问“PID 助手”时，缺省情况下显示**常规**选项卡。

描述

下表介绍了**常规**选项卡上的设置。

参数	描述
操作模式	代表要使用的 PID 模式： ● 未配置 ● PID ● AT + PID ● AT ● 字地址 有关操作模式的详细信息，请参阅 PID 操作模式 (参见第 *204* 页)。
字地址	您可在此文本框中提供内存字 (%MWxx)，以便通过编程来设置操作模式。根据要设置的操作模式，内存字可采用下列四个可能的值： ● %MWx = 0 （已禁用 PID） ● %MWx = 1 （仅设置 PID） ● %MWx = 2 （设置 AT + PID） ● %MWx = 3 （仅设置 AT） ● %MWx = 4 （仅设置 PI）
PID 状态	如果选中此框以启用该选项，则可在 PID 控制器使用的关联字段 (%MWxx) 中提供存储器字，以便在运行 PID 控制器和 / 或自调节功能时存储当前 PID 状态。有关详细信息，请参阅 PID 状态和检测到的错误码 (参见第 *236* 页)。

图形助手



图形助手有助于直观显示 PID 功能的构建方式。该动态图像随配置进行更新。

下列图标阐述了何时可对该图标进行访问，以及单击该图标时所出现的现象：

显示	描述
	单击此按钮以显示 **PID** 选项卡 (参见第 *227* 页) 的 " 设置点 " 字段。
	单击此按钮以显示 **PID** 选项卡 (参见第 *227* 页)。
	单击此按钮以显示**输出**选项卡 (参见第 *230* 页)。
	单击此按钮以显示**输入**选项卡 (参见第 *226* 页)。

显示	描述
	单击此按钮以显示 **AT** 选项卡 (参见第 *228* 页)。
	单击此按钮以显示 **AT** 选项卡 (参见第 *228* 页)。
	在**输入**选项卡 (参见第 *226* 页) 的**转换**区域中选中**授权**选项时，将出现该按钮。
	在**输入**选项卡 (参见第 *226* 页) 的**报警**区域中选中**授权**选项时，将出现该按钮。
	若**限制**与**输出**选项卡 (参见第 *230* 页) 中限制区域的约束条件不一致，则会出现该按钮。
	若手动模式与**输出**选项卡 (参见第 *230* 页) 中手动模式区域的约束条件不一致，则会出现该按钮。
	单击此按钮以显示**输出**选项卡 (参见第 *230* 页)。
	在**输出**选项卡 (参见第 *230* 页) 的 “ 输出 PWM” 区域中选中**授权**选项时，将出现该按钮。

输入选项卡

简介

本节介绍 PID 的**输入**选项卡。**输入**选项卡用于输入 PID 输入参数。

仅当处于离线模式以及从**常规**选项卡中选择某一操作模式后，方可访问此选项卡。

描述

下表描述了您可以定义的设置。

参数	描述	
测量	指定含有待控制的过程值的变量。 默认标度为 0 到 10,000。您可以输入内存字 (%MWxx) 或模拟量输入。	
转换	**授权**	选中此框以将过程值 **[0...10,000]** 转换到线性范围中 **[最小值 ... 最大值]**。
	最小值 **最大值**	指定转换标度的最小值和最大值。随后，过程值便会在 **[最小值 ... 最大值]** 区间内自动重新转换。 **最小值**或**最大值**可为内存字 (%MWxx)、常量字 (%KWxx) 或介于 -32768 和 +32767 之间的某个值。 **注意：最小值**必须小于**最大值**。
报警	**授权**	选中此框以在输入变量中激活报警。 报警值应根据转换阶段完成后得到的过程变量进行确定。转换激活时，报警值必须介于**最小值**到**最大值**之间。否则，报警值将介于 0 到 10,000 之间。
	低 **输出**	在**低**字段中指定低位报警值。 该值可为内存字 (%MWxx)、常量 (%KWxx) 或直接值。 **输出**必须包含位的地址 （当达到下限时该位将设为 1）。**输出**可为内存位 (%Mxx) 或输出。
	高 **输出**	在**高**字段中指定高位报警值。 该值可为内存字 (%MWxx)、常量 (%KWxx) 或直接值。 **输出**必须包含位的地址 （当达到上限时该位将设为 1）。**输出**可为内存位 (%Mxx) 或输出。

PID 选项卡

简介

PID 选项卡可用于输入内部 PID 参数。

仅当处于离线模式以及已从**常规**选项卡中选择某一操作模式后，方可访问该选项卡。

描述

下表描述了您可以定义的设置：

参数	描述	
设置点	指定 PID 设置点值。该值可为内存字 (%MWxx)、常量字 (%KWxx) 或直接值。 因此，当禁止转换时，该值必须介于 0 和 10,000 之间。否则，它必须介于用于转换的**最小值**和**最大值**之间。	
修正器类型	若先前已在 PID 属性表中选择 **PID** 或 **AT + PID** 操作模式，则可以从下拉列表中选择所需的校正器类型（**PID** 或 **PI**）。若已选择其他模式（**AT** 或 **字地址**），则**校正器类型**将设为**自动**并灰化（无法手动对其进行修改）。 若已从下拉列表中选择 **PI**， Td 参数将强制设为 0 并禁用该字段。	
参数 (1)	**Kp (x0,01s)**	指定 PID 比例增益，并乘以 100。 该值可为内存字 (%MWxx)、常量字 (%KWxx) 或直接值。 Kp 参数的有效范围为：0 < Kp < 10,000。 **注意：**若误将 Kp 设置为 0（Kp ≤ 0 时无效），则 PID 功能将自动赋以缺省值 Kp=100。
	Ti (x0,1s)	为 0.1 秒时基指定积分时间。 该值可为内存字 (%MWxx)、常量字 (%KWxx) 或直接值。 该值必须介于 0 到 36,000 之间。 **注意：**要禁用 PID 的积分操作，请将该系数设置为 0。
	Td (x0,1s)	指定 0.1 秒时基的微分时间。 该值可为内存字 (%MWxx)、常量字 (%KWxx) 或直接值。 该值必须介于 0 到 10,000。 **注意：**要禁用 PID 的微分操作，请将该系数设置为 0。
采样周期	在此处指定 10^{-2} 秒（10 毫秒）时基的 PID 采样周期。 该值可为内存字 (%MWxx)、常量字 (%KWxx) 或直接值。 该值必须介于 1（0.01 秒）到 10,000（100 秒）。	

(1) 启用自调节后，您便无需再设置 Kp、 Ti 和 Td 参数，因为自调节算法将以自动执行和编程方式自动对这些参数进行设置。在此情况下，您必须在这些字段中仅输入**内部字地址** (%MWxx)。启用自动调谐功能后请勿输入常量或直接值。

AT 选项卡

简介

AT 选项卡与自调节功能有关。有关更多详细信息，请参阅使用自调节进行 PID 调节 (参见第 *214* 页)。

仅当处于离线模式以及已从**常规**选项卡中选择某一操作模式后，方可访问该选项卡。

描述

PID 自调节是一个开放回路过程，它直接作用于控制过程，并只接受过程值 (PV) 限制和输出设置点提供的调节或任何限制。因此，必须在过程指定的允许范围内仔细选择这两个值，以避免潜在的过程过载。

⚠ 警告

PID 运行不稳定

- 必须在充分了解过程值 (PV) 极限和输出设定点值对于机器或者过程影响的条件下对其进行设置。
- 切勿超出过程值和输出设定点值的允许范围。

不遵循上述说明可能导致人员伤亡或设备损坏。

⚠ 警告

意外的设备操作

切勿将继电器输出与 PID 功能结合使用。

不遵循上述说明可能导致人员伤亡或设备损坏。

下表描述了您可以定义的设置：

字段	描述
授权	选中该框可启用自调节操作。 该复选框有两种使用方式，具体取决于您在 PID 功能的**常规**选项卡中手动设置操作模式还是通过字地址： ● 如果从**常规**选项卡 (参见第 *223* 页) 中将**操作模式**设置为 **PID + AT** 或 **AT**，则“授权”选项将自动选中且不可用 （无法清除）。 ● 如果通过字地址 %MWx (%MWx = 2: **PID + AT**; %MWx = 3: **AT**) 设置操作模式，则必须手动选中“授权”选项，这样才能配置自调节参数。
动态 AT 修正器	此参数允许您选择 AT 过程的动态。此参数影响 AT 过程计算的比例增益 (Kp) 值。
AT 触发器	此参数允许您每次在专用位 （存储器位或数字量输入位）上检测到上升沿时启动 AT 过程。

计算的 Kp、 Ti、 Td 系数

一旦完成自调节过程，计算的 Kp、 Ti 和 Td PID 系数将被存储到其各自的存储器字 (%MWx) 中。

输出选项卡

简介

该选项卡用于输入 PID 输出参数。

仅当处于离线模式以及已从**常规**选项卡中选择某一操作模式后，方可访问该选项卡。

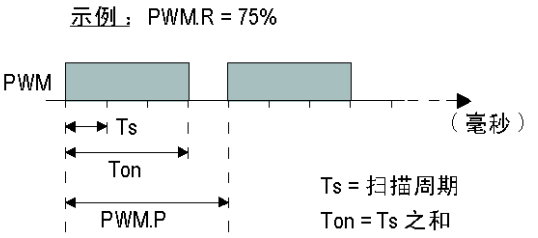
描述

下表描述了您可以定义的设置：

字段	描述
操作	在此指定在过程中执行的 PID 类型。可用选项有三个：**反向**、**直接**和**位地址**。如果增大输出导致过程值测量值增大，请定义反向操作 （“ 反向 ”）；反之，如果这导致过程值下降，请将 PID 设置为直接 （“ 直接 ”）。 如果选择**位地址** (1)，则可以通过修改关联位 （存储器位 (%Mxx) 或输入地址 (%Ix.y)）来修改操作类型。 如果操作为**直接**，则将内存位设置为 1；如果操作为**反向**，则将内存位设置为 0。
限制	指定是否要对 PID 输出施加限制。可用选项有 3 个：**启用**、**禁用**和**位地址**。 选择**启用**可将**位**设置为 1，选择**禁用**可将**位**设置为 0。 选择**位地址**则可通过修改关联位 （存储器位 (%Mxx) 或输入地址 (%Ix.y)）来进行位限制管理。 请设置 PID 输出的上限和下限。 **最小值**或**最大值**可为内存字 (%MWxx)、常量字 (%KWxx) 或介于 1 和 10,000 之间的某个值。 **注意**：**最小值**必须总是小于**最大值**。
手动模式	指定是否要将 PID 更改为手动模式。可用选项有 3 个：**启用**、**禁用**和**位地址**。 如果选择**位地址**，则可以修改关联位 （内存位 (%Mxx) 或输入），以便使用程序切换到手动模式 （位设为 1）或自动模式 （位设为 0）。 手动模式的**输出**必须包含您希望在 PID 处于手动模式 (参见第 *217* 页) 时分配给模拟量输出的值。此**输出**可以是字 (%MWxx) 或 [0...10,000] 格式的直接值。
模拟量输出	指定要在自调节模式时使用的 PID 输出。 此**模拟输出** (2) 可以是存储器字地址或模拟输出地址。

(1) 启用自调节时，自调节算法自动为控制过程确定正确的动作类型 （“ 直接 ” 或 “ 反向 ”）。然后，必须在相关的**位地址**文本框中仅输入一个存储器位 (%Mxx)。

(2) 输入存储器地址 (%MWxx) 或模拟输出地址 (%QWx.y)。

字段	描述
输出 PWM	选中此框可使用 PID 的 PWM 功能。 请在**周期 （0.1 秒）**文本框中指定调制周期。此周期必须是介于 1 到 500 之间的值，可以是内存字 (%MWxx) 或常量字 (%KWxx)。 PWM 精确度取决于 PWM 周期和扫描周期。当 PWM 比率 (%PWM.R) 有最多值时，该精确度会有所提升。例如，当扫描周围为 20 微秒且 PWM 周期为 200 微秒时， PWM.R 的值可为 0%、 10%、 20%、 30%、 40%、 50%、 60%、 70%、 80%、 90%、 100%。当扫描周围为 50 微秒且 PWM 周期为 200 微秒时， PWM.R 的值可为周期 PWM.P 的 0%、 25%、 50%、 75% 和 100%。  指定 PWM 输出位作为**输出**中的值。该值可以是内存位 (%Mxx) 或输出地址。有关 PWM 功能的详细信息，请参阅 Logic Controller 的*功能库指南*。
(1) 启用自调节时，自调节算法自动为控制过程确定正确的动作类型 （“ 直接 ” 或 “ 反向 ”）。然后，必须在相关的**位地址**文本框中仅输入一个存储器位 (%Mxx)。 **(2)** 输入存储器地址 (%MWxx) 或模拟输出地址 (%QWx.y)。	

节 6.5
PID 编程

使用 PID 功能

本节介绍如何使用 **PID** 功能并提供其编程指南。

本节包含了哪些内容？

本节包含了以下主题：

描述

简介

比例 - 积分 - 微分 (PID) 是一个广泛用于工业控制系统的常规控制回路反馈机制（控制器）。 PID 控制器使用一个涉及 3 个独立常量参数（分别以 P、I 和 D 表示的比例、积分和微分值）的算法。

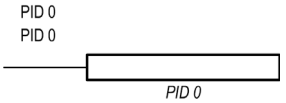
主要功能

SoMachine Basic PID 功能的主要功能如下所示：

- 模拟量输入
- 可配置测量值的线性转换
- 高位或低位可配置输入报警
- 模拟量或 PWM 输出
- 可配置输出的切断
- 可配置的直接或反向操作
- 自调节功能

示意图

这是 SoMachine Basic 的梯形图编辑器中的 PID 功能：



注意：PID 和 PID 编号之间必须有空格 （例如， PID< 空格 >0）。

参数

与 Timer 或 Counter 功能块不同， SoMachine Basic 中没有 PID 功能块。指令 [PID x] 仅支持 PID 控制回路功能，其中 x 是 PID 编号。

要配置 PID 功能，请转至**编程**窗口，单击**工具 →PID**，然后编辑 PID 属性 （请参阅下表了解配置参数）。

PID 功能具有以下参数：

参数	描述	值
已使用	如果在项目的任意位置使用了 I/O，则选中该参数	True/False False （缺省值）
PID	当前 PID 对象的名称	程序只能包含数量有限的 PID 功能。有关 Logic Controller 可用的最大 PID 对象数，请参阅硬件平台的*编程指南*。
符号	当前 PID 对象的符号	与此 PID 对象关联的符号。有关详细信息，请参见定义符号 *(参见 SoMachine Basic, Operating Guide)*。
[...]	用于启动助手的按钮	单击可显示 **PID 助手**屏幕。有关详细信息，请参阅 PID 助手 (参见第 *221* 页)。
注释	注释	可以将注释与此对象进行关联。

编程和配置

简介

本节介绍了如何对 SoMachine Basic PID 控制器进行编程和配置。

启用 PID 控制器

以下示例在位 %M0 设置为 1 的情况下启用 PID 0 控制器循环：

梯级	指令
0	LD %M0 [PID 0]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

PID 模拟测量

PID 功能使用模拟测量和设置点来完成 PID 纠正，并以相同的格式生成模拟指令或在数字量输出中生成 PWM。

要以全标度 （最高分辨率）使用 PID，请以 [0...10,000] 格式配置 PID 控制器测量专用的模拟量输入。但是，如果使用缺省配置 [0...4095]， PID 控制器仍将正常运行。

配置扫描周期

在使用 SoMachine Basic PID 控制器时，您必须将 Logic Controller 的扫描模式配置为**周期**扫描模式 （**程序**选项卡，**任务 →主任务**）。在周期扫描模式下， Logic Controller 均会以定期时间间隔开始每次扫描，以便使整个测量期间的采样率都一致。有关配置扫描模式的详细信息，请参阅 *SoMachine 基本操作指南*。

在周期扫描模式下，如果 Logic Controller 的扫描时间超过用户程序定义的周期，则系统会将系统位 %S19 设置为 1。

PID 状态和检测到的错误代码

简介

SoMachine Basic PID 控制器能够将 PID 控制器和自调节过程的当前状态写入到用户定义的存储器字中。有关如何启用和配置 PID 状态存储器字的详细信息，请参阅 PID 助手 (参见第 *221* 页) 的**常规**选项卡 (参见第 *223* 页)。

PID 状态存储器字可以记录以下类型的 PID 信息：

- PID 控制器的当前状态
- 自调节过程的当前状态
- PID 检测到的错误代码
- 自调节检测到的错误代码

注意： PID 状态存储器字处于只读状态。

PID 状态存储器字

PID 状态	描述
0000 （十六进制）	PID 控制不处于活动状态
2000 （十六进制）	PID 控制正在进行
4000 （十六进制）	已到达 PID 设置点

自调节状态存储器字

自调节状态	描述
0100 （十六进制）	自调节阶段 1 (参见第 *215* 页) 正在进行
0200 （十六进制）	自调节阶段 2 (参见第 *215* 页) 正在进行
0400 （十六进制）	自调节阶段 3 (参见第 *215* 页) 正在进行
0800 （十六进制）	自调节阶段 4 (参见第 *215* 页) 正在进行
1000 （十六进制）	自调节阶段完成

PID 检测到的错误代码

下表介绍了控制 PID 期间可能出现的检测到的错误：

检测到的错误代码	描述
8001 （十六进制）	操作模式值超出范围
8002 （十六进制）	线性转换的最大最小值相等
8003 （十六进制）	离散量输出的上限低于下限
8004 （十六进制）	过程值限制超出线性转换范围
8005 （十六进制）	过程值限制小于 0 或大于 10000
8006 （十六进制）	设置点超出线性转换范围
8007 （十六进制）	设置点小于 0 或大于 10000
8008 （十六进制）	控制动作与自调节开始时确定的动作不一致

自调节检测到的错误代码

下表记录了自调节检测到的错误消息，并介绍了可能的原因以及故障排除操作：

检测到的错误代码	描述
8009 （十六进制）	已达到过程值 (PV) 限制。由于自调节是一个开放回路过程，因此过程值 (PV) 限制将用作最大允许值。
800A （十六进制）	采样周期过短，或输出设置点过低。拉长采样周期或增大自调节输出设置点值。
800B （十六进制）	Kp 为零。
800C （十六进制）	时间常量为负，因此采样周期可能过长。有关详细信息，请参阅自调节使用限制 (参见第 *251* 页)。
800D （十六进制）	延迟为负。
800E （十六进制）	计算 Kp 时检测到错误。自调节算法不稳定 （未整合）。这可能是由于： ● 自调节期间的过程干扰使得过程静态增益评估失真。 ● 过程值瞬时响应不够大，无法进行自调节来确定静态增益。 ● 综上所述。 检查 PID 和自调节参数并进行调整，以改善整合。同时检查有无可能影响过程值的干扰。尝试修改： ● 输出设置点 ● 采样周期 请确保自调节过程中无过程干扰。
800F 十六进制	时间常量超过延迟比率， $\tau/\theta > 20$。 PID 调节或许不再稳定。有关更多详细信息，请参阅自调节使用限制 (参见第 *251* 页)。
8010 （十六进制）	时间常量超过延迟比率， $\tau/\theta < 2$。 PID 调节或许不再稳定。有关详细信息，请参阅自调节使用限制 (参见第 *251* 页)。

检测到的错误代码	描述
8011（十六进制）	已超过静态增益 Kp 的限值， Kp>10,000。某些应用程序变量的测量灵敏度可能太低。范围必须重新调整在 [0...10,000] 区间之内。
8012（十六进制）	已超过积分时间常量 Ti 的计算值， Ti > 20,000。
8013（十六进制）	已超过微分时间常量 Td 的计算值， Td > 10,000。

章 7
时钟功能

概述

本节介绍控制器的时间管理功能。

本章包含了哪些内容？

本章包含了以下主题：

时钟功能

简介

在配有实时时钟 (RTC) 功能的 Logic Controller 上，用户可在 SoMachine Basic 连接到 Logic Controller 时使用下列实时时钟功能：

- **计划**功能块 (参见第 *197* 页) 可用于在预定义或计算所得的时间控制操作。
- **时间 / 日期戳** (参见第 *241* 页) 用于向事件分配时间和日期并测量事件的持续时间。

可通过程序设置 (参见第 *241* 页) 实时时钟。当控制器关闭时，控制器电池将帮助时钟设置继续操作长达 1 年。控制器未配备充电电池。电池的平均寿命应为 4 年，并且应在寿命结束前更换。为了不在更换电池期间丢失数据，请在从控制器中取出电池后 120 秒内更换电池。

实时时钟采用 24 小时格式，并将闰年考虑在内。

时间和日期戳

简介

系统字 %SW49 到 %SW53 包含 BCD 格式的当前日期和时间，对于在外设上显示或向外设传输十分有用。这些系统字可用于存储事件的时间和日期。

BTI 指令用于将 BCD 格式的日期和时间转换为二进制格式。有关详细信息，请参阅二进码十进制 / 二进制转换指令 (参见第 *72* 页)。

记录事件日期

要将日期与事件相关联，请使用分配操作将系统字的内容传输到内部字，然后处理这些内存字（例如，使用 EXCH 指令传输到显示单元）。

编程示例

下列示例显示了如何记录输入 %I0.1 的上升沿的日期：

梯级	指令
0	LDR %I0.1 [%MW11:5:=%SW49:5]

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

检测到事件后，字表将包括以下内容：

编码	最高有效字节	最低有效字节
%MW11	-	星期几 (1)
%MW12	00	秒
%MW13	时	分
%MW14	月	日
%MW15	世纪	年
(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

2013 年 6 月 3 日星期一 13:40:30 的示例数据：

字	值（十六进制）	含义
%MW11	0001	星期一
%MW12	0030	30 秒
%MW13	1340	13 时， 40 分
%MW14	0603	06 = 6 月 3 日
%MW15	2013	2013

上次停机的日期和时间

系统字 %SW54 到 %SW57 包含上次停机的日期和时间，而字 %SW58 包含表明上次停机原因的 BCD 格式的代码。

设置日期和时间

简介

通过使用以下方法之一可以更新日期和时间设置：

- SoMachine Basic
  用户可在用于设置 Logic Controller 时间的两种模式之间进行选择：
  - 手动：该模式可显示日期 / 时间选择工具，允许用户手动选择要在 Logic Controller 内设置的时间。
  - 自动：该模式可显示运行有 SoMachine Basic 且用于在逻辑控制器内设置时间的 PC 的当前时间。

  （有关详细信息，请参阅 SoMachine Basic 操作指南。）
- 系统字
  使用系统字 %SW49 到 %SW53 或系统字 %SW59。

注意：当逻辑控制器中的 RTC 功能可用时，用户可设置日期和时间 （请参阅逻辑控制器的*编程指南*）。

使用 %SW49 到 %SW53

要使用系统字 %SW49 到 %SW53 以设置日期和时间，必须将位 %S50 设为 0。一旦设置了日期和时间，位 %S51 必须设置为 1。这将产生以下结果：

- 通过内部时钟取消字 %SW49 到 %SW53 的更新。
- 将写入字 %SW49 到 %SW53 中的值传输到内部时钟。

下表列出了包含表示实时时钟 (RTC) 功能的当前日期和时间值 （以 BCD 形式）的系统字：

系统字	描述
%SW49	星期的第 xN 天 （N=1 为星期一）
%SW50	00SS：秒
%SW51	HHMM：小时和分钟
%SW52	MMDD：月份和日期
%SW53	CCYY：世纪和年

有关系统位和系统字的完整列表，请参阅控制器的*编程指南*。

编程示例：

梯级	指令	注释
0	LD %S50 R %S50	–
1	LD %I0.1 [%SW49:=%MW10] [%SW50:=%MW11] [%SW51:=%MW12] [%SW52:=%MW13] [%SW53:=%MW14] S %S50	请参阅二进码十进制 / 二进制转换指令 (参见第 *72* 页)。

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

字 %MW10 到 %MW14 包含新日期和时间（请参见 BCD 码概述 (参见第 *72* 页)）并对应于字 %SW49 到 %SW53 的编码。

字表必须包含新的日期和时间：

编码	最高有效字节	最低有效字节
%MW10	–	星期几 (1)
%MW11	–	秒
%MW12	时	分
%MW13	月	日
%MW14	世纪	年
(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		

2013 年 6 月 3 日星期一的示例数据：

字	值 （十六进制）	含义
%MW10	0001	星期一
%MW11	0030	30 秒
%MW12	1340	13 时， 40 分
%MW13	0603	06 = 6 月 3 日
%MW14	2013	2013

使用 %SW59

更新日期和时间的另一种方法是使用系统位 %S59 和日期调整系统字 %SW59。

将位 %S59 设置为 1，可通过字 %SW59 调整当前的日期和时间。 %SW59 可在上升沿对每个日期和时间部分进行递增或递减。

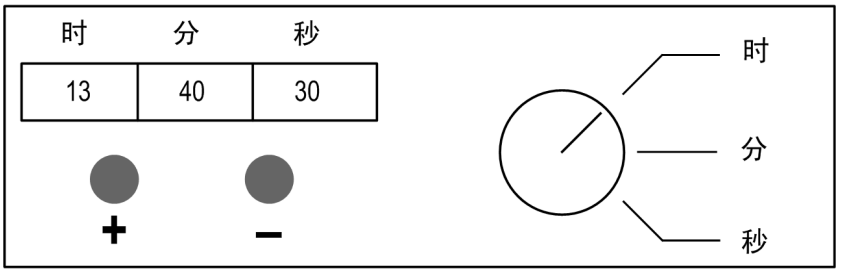
下表介绍了系统字 %SW59 的每个位，用于调整日期和时间参数：

递增	递减	参数
位 0	位 8	星期几 (1)
位 1	位 9	秒
位 2	位 10	分
位 3	位 11	时
位 4	位 12	日
位 5	位 13	月
位 6	位 14	年
位 7	位 15	世纪 (1)
(1) 用户不得修改星期几和世纪 （增量或减量）。		

有关系统位和系统字的完整列表，请参阅控制器的*编程指南*。

应用示例

下列前面板可用于修改内部时钟的时、分和秒。



命令的描述：

- 时 / 分 / 秒开关分别使用输入 %I0.2、 %I0.3 和 %I0.4 选择要更改的时间显示。
- 按钮 “+” 使用输入 %I0.0 以递增所选的时间显示。
- 按钮 “-” 使用输入 %I0.1 以递减所选的时间显示。

下列程序将从面板读取输入并设置内部时钟：

梯级	指令	注释
0	LD %M0 ST %S59	–
1	LD %I0.2 ANDR %I0.0 ST %SW59:X3	时
2	LD %I0.2 ANDR %I0.1 ST %SW59:X11	–
3	LD %I0.3 ANDR %I0.0 ST %SW59:X2	分
4	LD %I0.3 ANDR %I0.1 ST %SW59:X10	–
5	LD %I0.4 ANDR %I0.0 ST %SW59:X1	秒
6	LD %I0.4 ANDR %I0.1 ST %SW59:X9	–

注意：请参阅可转换性过程 (参见第 *14* 页) 以获取等效梯形图。

附录

附录 A
PID 参数

本章包含了哪些内容?

本章包含了以下主题:

PID 参数的作用和影响

简介

本节对 PID 参数的作用和影响进行了描述。

PID 控制器模型

SoMachine Basic PID 控制器实现了混合（串行 - 并行）PID 修正。积分和微分操作将以独立和并行方式执行。比例操作会对积分和微分操作的组合输出的产生作用。

计算算法

根据积分时间常量 (Ti) 的值，可以使用两种不同的计算算法：

- 如果 Ti ≠ 0，使用增量算法，
- 如果 Ti = 0，使用位置算法及应用于 PID 输出的 +5000 偏移。

操作的影响

比例动作用于影响过程响应速度。比例动作增加表示：

- 响应更快
- 静态误差更少
- 稳定性降低

积分动作用于消除静态误差。积分动作增加 （即积分时间 Ti 缩短）表示：

- 响应更快
- 稳定性降低

微分动作是可预见的。实际上，它有考虑到偏差中的变化速度 （这样就能通过在偏差增大时加快过程响应时间并在偏差减小时减慢过程响应时间来参与更改操作）。微分动作增加 （即积分时间增加）代表：

- 响应更慢
- 过冲减少

注意：从微分时间的角度考虑， Td 是用于参与微分变化的时间。 Td 值太低或太高都会导致有害振荡。

必须针对各个动作在速度和稳定性之间找到适当的折中方案。

PID 控制回路的限制

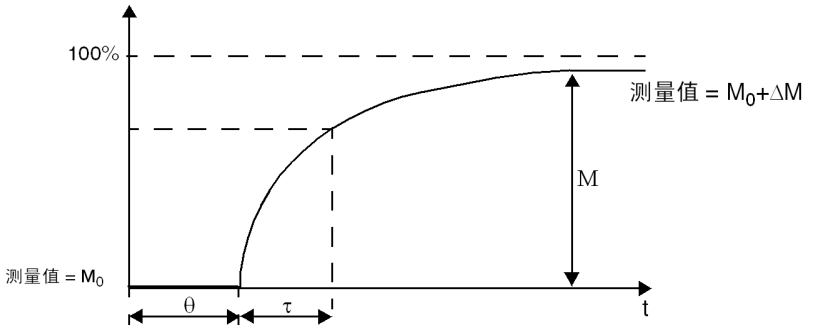
过程被吸收到具有转移功能的一阶纯延迟：

$$H(p) = K \times \frac{e^{-\theta p}}{1+\tau p}$$

其中：

τ：模型时间常量

θ：模型延迟



过程控制性能取决于比率 $\frac{\tau}{\theta}$

在以下域中可以实现适当的 PID 过程控制：$2 < \frac{\tau}{\theta} < 20$

PID 过程控制最适用于满足以下条件的过程调节：

- 对于 $\frac{\tau}{\theta} < 2$，也就是说，对于快速控制回路（低 θ）或对于具有大延迟（高 t）的过程而言，PID 过程控制并不适用。在此类情况下，应使用更复杂的算法。
- 对于 $\frac{\tau}{\theta} > 20$，使用阈值加滞后值的过程控制就足够了。

PID 参数调整方法

简介

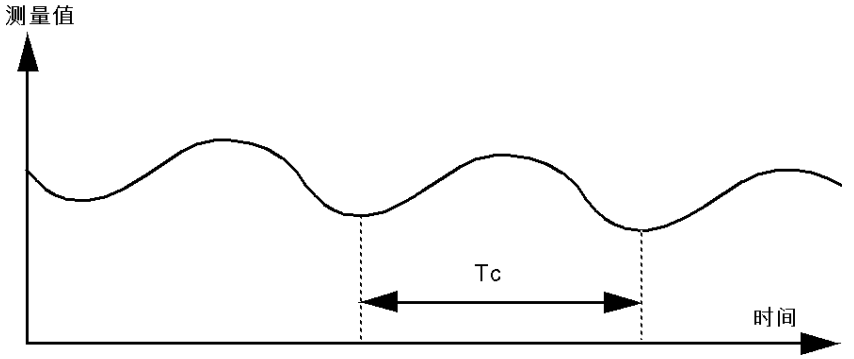
调整 PID 参数的方法有很多种。首选方法是 Ziegler 和 Nichols，它们具有 2 个变体：

- 闭合回路调整
- 开放回路调整

在实现其中任一方法之前，必须设置 PID 动作 (参见第 *230* 页)。

闭合回路调整

该原理是使用比例命令 （Ti = 0、 Td = 0），从而通过增大比例系数来启动过程，直到过程在将级别应用于 PID 修正器设定点后再次开始振荡为止。需要提高导致非衰减振荡的关键比例增益 (Kpc) 和振荡周期 (Tc)，以降低实现最佳调节的值。



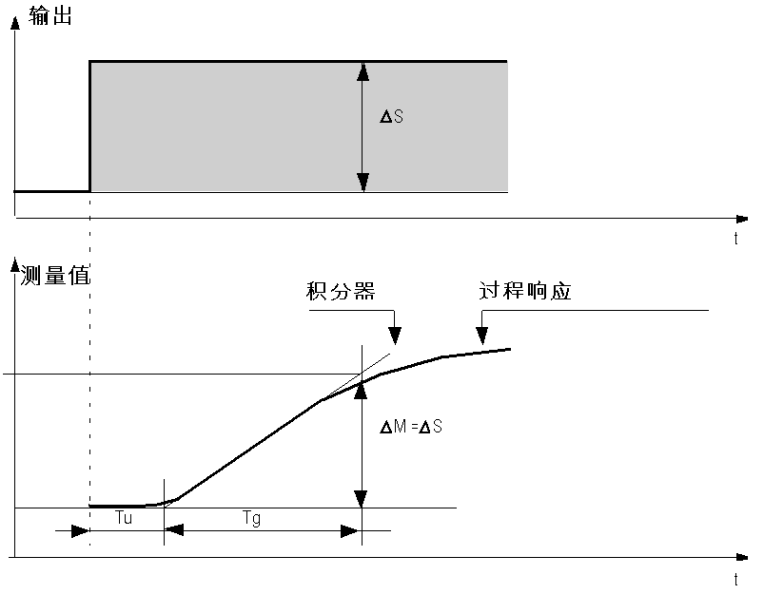
基于所使用的修正器类型 （PID 或 PI），系数调整将通过以下值来执行：

修正器	Kp：比例增益	Ti：积分时间	Td：微分
PID	Kpc/1.7	Tc/2	Tc/8
PI	Kpc/2.22	0.83 x Tc	–

通过自调节实现 PID 时，**动态 AT 修正器**参数将影响比例增益 (Kp) 值。在 AT 过程中计算比例增益取决于选择的动态修正器速度。选择**快速**将提供快速响应并带有更多过冲，而选择**慢速**则提供较慢的响应时间并带有较少的过冲。

开放回路调整

由于调节器处于手动模式 (参见第 *217* 页)，因此您将电平施加于输出，并使过程响应的启动方式与具有纯延迟时间的积分器相同。



右手边的交点代表具有时间轴的积分器，该交点确定时间 Tu。接下来， Tg 时间定义为所控制变量 （测量值）获得与调节器输出相同的变化幅度 （标度的百分比）所需的时间。

基于所使用的修正器类型 （PID 或 PI），系数调整将通过以下值来执行：

修正器	Kp：比例增益	Ti：积分时间	Td：微分
PID	-1.2 Tg/Tu	2 x Tu	0.5 x Tu
PI	-0.9 Tg/Tu	3.3 x Tu	–

注意：有关参数单位的详细信息，请参阅 **PID** 选项卡 (参见第 *227* 页)。

该调整方法会提供一个非常动态的命令，这体现在设置点脉冲变化期间会发生有害过冲。在这种情况下，请降低比例增益，直到获得所需行为为止。此方法不需要任何关于过程的本质和顺序的假设。您可以将其应用于稳定的过程，就像将其应用到真实的积分过程一样。对于慢速过程 （例如，玻璃行业），用户只需要响应的开端就能调节系数 Kp、 Ti 和 Td。

术语

%

根据 IEC 标准，% 是标识 Logic Controller 中用于存储程序变量、常量和 I/O 等值的内部存储器地址的前缀。

%KW

根据 IEC 标准， %KW 表示常量字。

%MW

根据 IEC 标准， %MW 表示存储器字寄存器 （例如，存储器字类型的语言对象）。

%Q

根据 IEC 标准， %Q 表示输出位 （例如，数字量输出类型的语言对象)。

功能块

拥有一个或多个输入并返回一个或多个输出的编程单元。 FBs 通过实例 （具有专用名称和变量的功能块副本）进行调用，且每个实例在从一个调用到另一个调用会保持原有状态 （输出和内部变量）。

示例：定时器、计数器

指令列表语言

以指令列表语言编写的程序，包括由控制器按顺序执行的一系列基于文本的指令。每个指令均包括一个行号、一个指令代码和一个操作数 （请参阅 IEC 61131-3）。

梯形图语言

控制器程序指令的图形表示方法，其中包括控制器按顺序执行的一系列梯级中的触点、线圈和功能块符号 （请参阅 IEC 61131-3）。

模拟量输入

用于将收到的电压或电流电平转换为数值。可以在 Logic Controller 中存储和处理这些值。

模拟量输出

在 Logic Controller 内转换数值，并按比例发送电压或电流电平。

索引