SIEMENS

SIMATIC

S7
S7-1200 可编程控制器

系统手册

11/2009
A5E02486685-02

快速接线模块：控制柜装配的全新概念

常规接线模式

快速接线模式

全球独家推出 全覆盖型省配线解决方案

接线端子柜 继电器柜

隔离器柜 安全栅柜

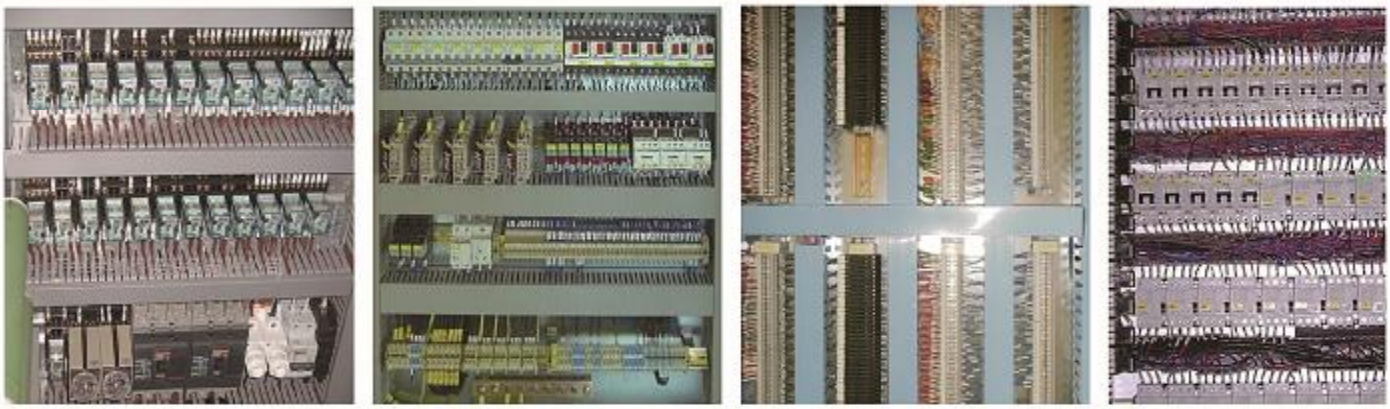
快速接线模块：控制柜装配的全新概念

传统接线模式

密密麻麻的电器元件：接线端子、保险、继电器、电源开关

塞满线槽的的配线：元件之间大量的连接导线，离开图纸无从下手

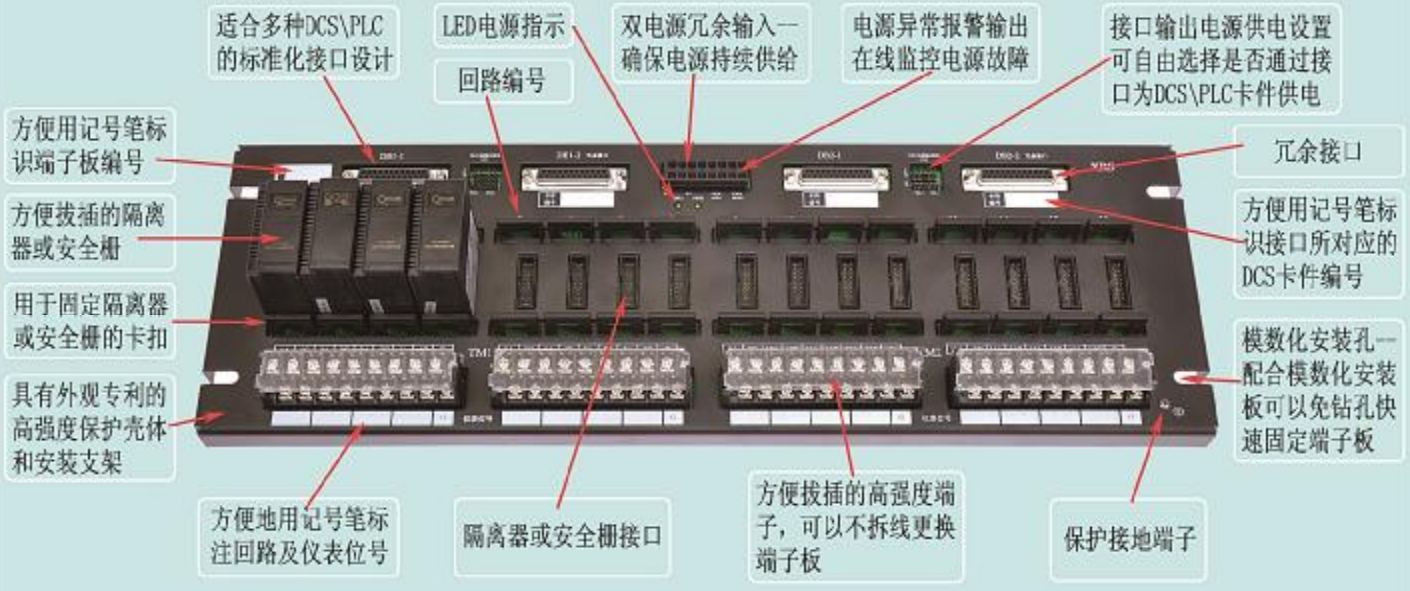
繁琐的工序：设计选型、配件采购、打孔、元件固定、配线、测试、整理



传统接线方式占空间大，导线多而杂乱，连接费时费力，出错不容易检修

适应现代工业的集成化模块产品

快速接线模式



领先的技术 贴心的设计

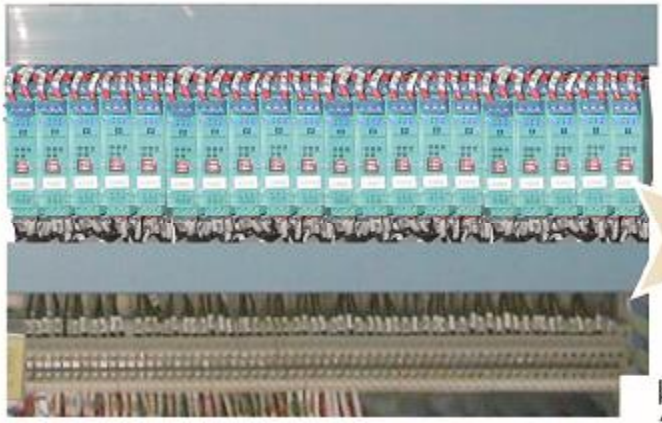
快速接线模块：控制柜装配的全新概念

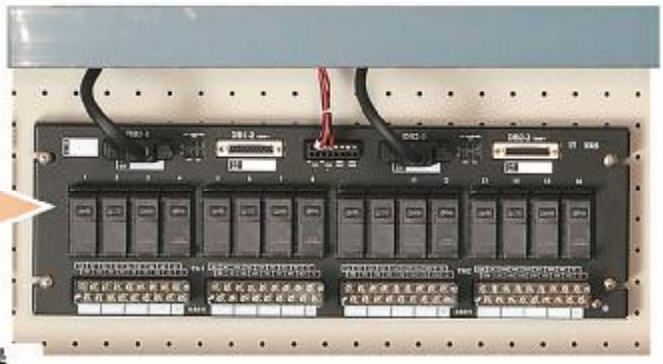
传统接线模式

快速接线模式

控制组件

继电器隔离





隔离器
安全栅

非隔离

- 一页简单表格替代配线蓝图
- 改变用途只需改动设置，为设计变更提供最大便利
- 让普工具备专业配线技工的装配水平
- 免拆线测试信号电流和故障指示功能轻松解决日常维护难题
- 丢掉米尺、电钻、打号机、压线钳、万用表和各式配件，一把螺丝刀搞定控制柜装配

快速接线模块：控制柜装配的全新概念

专利产品

多： 功能多、用途广　　**快：** 设计快、施工快

好： 美观好用、可靠　　**省：** 省人工、省材料

- 全系列模块化结构：将控制柜内接线附件设计成模块化结构的系列产品，包括信号输入输出、电源分配等，采用标准的安装尺寸，元器件透明化布置，指示一目了然，既美观又快捷。
- 功能全面：兼有信号隔离和驱动、本安保护、回路供电、卡件供电、信号指示、报警指示、回路保护、电源保护、信号转接等多项功能，全面提升配线质量。
- 简化设计和布线：采用板上跳线的型式解决了外部设备与系统卡件接线的差异化和复杂化问题。信号传输和供电巧妙有机的融合在一起，同一个卡件可以接驳不同类型的仪表和信号，只需改变板上跳线，减少了线路节点。盘内施工图纸简化为简单表格，降低了复杂线路对图纸的依赖性，彻底简化了施工图纸设计和盘内布线过程。
- 省略端子排：外部接线端配备的拔插式接线端子块可方便端子维修，更换端子板可不拆除接线。弹簧端子则具有接线快捷、压接可靠、故障率低的特点。1~2.5mm²导线可直接接驳端子板，不必再专门配备接线端子排。内部接线端采用专用接口，使用特制电缆与PLC或DCS快速插接。
- 全方位保护、不拆线停表、省略配电开关，避免误操作：各信号回路均配备了拔插方便的保险以提供全方位的安全保障，用户不必再为外部设备单独配备配电开关和保险端子。特别解决了DCS及PLC系统外围仪表停表的问题，检修、拆除外部设备只要拔下保险而不必拆线，既快捷高效又安全可靠防止错接。电源回路采用过压、过流、反接保护措施，确保系统设备安全运行。
- LED指示、不拆线测电流，方便维护维修：电源回路和信号回路均配备全方位的LED信号指示及保险熔断报警，配合巧妙的不拆线测量信号电流技术，测量信号电流只要将电流表表笔插入测试孔即可，整个测量过程设备不断电、不影响正常测控过程，为维护工作提供了极大方便。
- 快速装配：使用本公司设计的带标准模数孔的专用安装背板，可以抛开钻孔工具，只需一把螺丝刀就能完成柜内电气元件的安装。

广泛兼容

兼容国内外主流品牌DCS\PLC控制系统

SIEMENS 西门子　YOKOGAWA 横河电机　Honeywell 霍尼韦尔　ABB　EMERSON. 艾默生　FOXBORO 福克斯波罗　EUROTHERM 欧陆　Rockwell Automation Authorized Distributor Allen-Bradley

MTL Instruments　OMRON　三菱电机 MITSUBISHI ELECTRIC　Schneider Electric 施耐德电气　富士电机 Fuji Electric　LG 产电　FATEK 永宏

ADVANTECH 研华科技　SUPCON 浙大中控　和利时 HollySys　浙大中自　UNITEK 优稳　PHOENIX CONTACT 菲尼克斯　Weidmüller 魏德米勒　HITACHI Inspire the Next 日立

全面覆盖

涵盖非隔离、继电器隔离、隔离器、本安防爆四大类信号传输方式


快速接线模块方式
端子柜
继电器柜
滨州新大新

法律资讯

警告提示系统

为了您的人身安全以及避免财产损失，必须注意本手册中的提示。人身安全的提示用一个警告三角表示，仅与财产损失有关的提示不带警告三角。警告提示根据危险等级由高到低如下表示。

⚠危险
表示如果不采取相应的小心措施，**将会**导致死亡或者严重的人身伤害。

⚠警告
表示如果不采取相应的小心措施，**可能**导致死亡或者严重的人身伤害。

⚠小心
带有警告三角，表示如果不采取相应的小心措施，可能导致轻微的人身伤害。

小心
不带警告三角，表示如果不采取相应的小心措施，可能导致财产损失。

注意
表示如果不注意相应的提示，可能会出现不希望的结果或状态。

当出现多个危险等级的情况下，每次总是使用最高等级的警告提示。如果在某个警告提示中带有警告可能导致人身伤害的警告三角，则可能在该警告提示中另外还附带有可能导致财产损失的警告。

合格的专业人员

本文件所属的产品/系统只允许由符合各项工作要求的**合格人员**进行操作。其操作必须遵照各自附带的文件说明，特别是其中的安全及警告提示。 由于具备相关培训及经验，合格人员可以察觉本产品/系统的风险，并避免可能的危险。

按规定使用 Siemens 产品

请注意下列说明：

⚠警告
Siemens 产品只允许用于目录和相关技术文件中规定的使用情况。如果要使用其他公司的产品和组件，必须得到 Siemens 推荐和允许。正确的运输、储存、组装、装配、安装、调试、操作和维护是产品安全、正常运行的前提。必须保证允许的环境条件。必须注意相关文件中的提示。

商标

所有带有标记符号 ® 的都是西门子股份有限公司的注册商标。标签中的其他符号可能是一些其他商标，这是出于保护所有者权利的 目地由第三方使用而特别标示的。

责任免除

我们已对印刷品中所述内容与硬件和软件的一致性作过检查。然而不排除存在偏差的可能性，因此我们不保证印刷品中所述内容与硬件和软件完全一致。印刷品中的数据都按规定经过检测，必要的修正值包含在下一版本中。

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
德国

文件订购号: 6ES7298-8FA30-8KH0
℗ 10/2009

前言

手册用途

S7-1200 系列是一款可编程逻辑控制器 (PLC, Programmable Logic Controller)，可以控制各种自动化应用。 S7-1200 设计紧凑、成本低廉且具有功能强大的指令集，这些特点使它成为控制各种应用的完美解决方案。 S7-1200 型号和基于 Windows 的编程工具提供了解决自动化问题时需要的灵活性。

本手册提供了有关 S7-1200 PLC 的安装和编程信息，其主要用户是具备可编程逻辑控制器基本知识的工程师、编程人员、安装人员和电工人员。

所需的基本知识

要理解本手册，需要具备自动化和可编程逻辑控制器的基本知识。

手册适用范围

本手册适用于 STEP 7 Basic V10.5 和 S7-1200 产品系列。 有关本手册中所述 S7-1200 产品的完整列表，请参见技术规范 (页 321)。

证书、CE 标签、C 标记和其它标准

请参见技术规范 (页 321)以获取更多信息。

服务与支持

除了文档之外，我们还在 Internet 的以下网址处提供了专业技术知识：

http://www.siemens.com/automation/support-request

如需要回答任何技术问题、培训或订购 S7 产品方面的帮助，请与西门子经销商或销售部联系。 因为西门子销售代表都经过技术培训并掌握有关操作、过程和工业以及有关您使用的各种西门子产品的最具体的知识，所以他们能够最快最高效地回答您可能遇到的任何问题。

目录

1 产品概述

1.1 S7-1200 PLC 简介

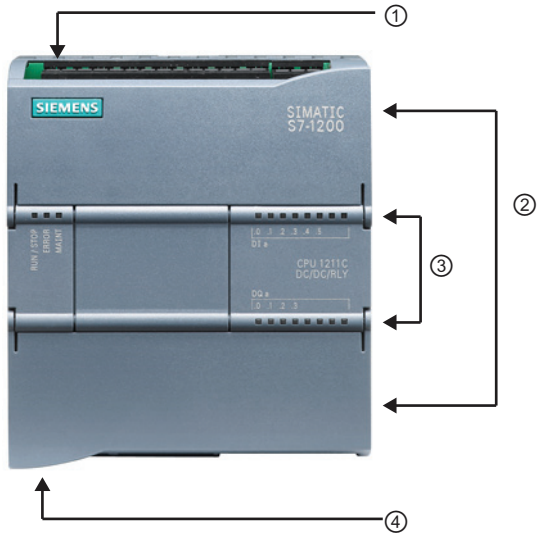
S7-1200 可编程逻辑控制器 (PLC, Programmable Logic Controller) 提供了控制各种设备以满足您自动化需要的灵活性和强大功能。 S7-1200 设计紧凑、组态灵活且具有功能强大的指令集，这些特点的组合使它成为控制各种应用的完美解决方案。

CPU 将微处理器、集成电源、输入电路和输出电路组合到一个设计紧凑的外壳中以形成功能强大的 PLC。 在您下载用户程序后，CPU 将包含监控应用中的设备所需的逻辑。 CPU 根据用户程序逻辑监视输入并更改输出，用户程序可以包含布尔逻辑、计数、定时、复杂数学运算以及与其它智能设备的通信。

有多种安全功能可用于保护对 CPU 和控制程序的访问：

- 每个 CPU 都提供密码保护功能，用户通过它可以组态对 CPU 功能的访问。
- 可以使用“专有技术保护”隐藏特定块中的代码。 有关详细信息，请参见“编程概念 (页 97)”一章。

CPU 提供一个 PROFINET 端口用于通过 PROFINET 网络通信。 还可使用通信模块通过 RS485 或 RS232 网络通信。



① 电源接口

② 可拆卸用户接线连接器（保护盖下面）

② 存储卡插槽（上部保护盖下面）

③ 板载 I/O 的状态 LED

④ PROFINET 连接器（CPU 的底部）

不同的 CPU 型号提供了各种各样的特征和功能，这些特征和功能可帮助用户针对不同的应用创建有效的解决方案。 有关特定 CPU 的详细信息，请参见技术规范 (页 321)。

特征	CPU 1211C	CPU 1212C	CPU 1214C
物理尺寸 (mm)	90 x 100 x 75		110 x 100 x 75
用户存储器 • 工作存储器 • 装载存储器 • 保持性存储器	 • 25 KB • 1 MB • 2 KB		 • 50 KB • 2 MB • 2 KB
本地板载 I/O • 数字量 • 模拟量	 • 6 点输入/4 点输出 • 2 路输入	 • 8 点输入/6 点输出 • 2 路输入	 • 14 点输入/10 点输出 • 2 路输入
过程映像大小	1024 字节输入 (I) 和 1024 字节输出 (Q)		
位存储器 (M)	4096 个字节		8192 个字节
信号模块扩展	无	2	8
信号板	1		
通信模块	3（左侧扩展）		
高速计数器 • 单相 • 正交相位	3 • 3 个，100 kHz • 3 个，80 kHz	4 • 3 个，100 kHz 1 个，30 kHz • 3 个，80 kHz 1 个，20 kHz	6 • 3 个，100 kHz 3 个，30 kHz • 3 个，80 kHz 3 个，20 kHz
脉冲输出	2		
存储卡	SIMATIC 存储卡（选件）		
实时时钟保持时间	通常为 10 天/40 摄氏度时最少 6 天。		
PROFINET	1 个以太网通信端口		
实数数学运算执行速度	18 µs/指令		
布尔运算执行速度	0.1 µs/指令		

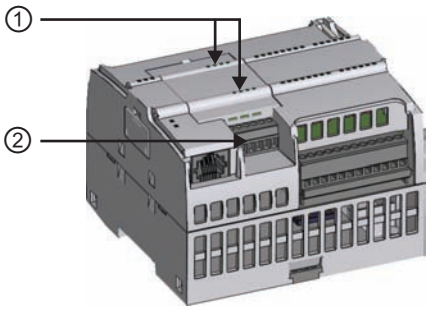
S7-1200 系列提供了各种信号模块和信号板用于扩展 CPU 的能力。 还可以安装附加的通信模块以支持其它通信协议。 有关特定模块的详细信息，请参见技术规范 (页 321)。

模块		仅输入	仅输出	输入/输出组合
信号模块 (SM)	数字量	8 x DC 输入	8 x DC 输出 8 x 继电器输出	8 x DC 输入/8 x DC 输出 8 x DC 输入/8 x 继电器输出
		16 x DC 输入	16 x DC 输出 16 x 继电器输出	16 x DC 输入/16 x DC 输出 16 x DC 输入/16 x 继电器输出
	模拟量	4 x 模拟量输入 8 x 模拟量输入	2 x 模拟量输出 4 x 模拟量输出	4 x 模拟量输入/2 x 模拟量输出
信号板 (SB)	数字量	-	-	2 x DC 输入/2 x DC 输出
	模拟量	-	1 x 模拟量输出	-
通信模块 (CM) • RS485 • RS232				

1.2 信号板

通过信号板 (SB, Signal Board) 可以给 CPU 增加 I/O。 可以添加一个具有数字量或模拟量 I/O 的 SB。SB 连接在 CPU 的前端。

- 具有 4 个数字量 I/O（2 x DC 输入和 2 x DC 输出）的 SB
- 具有 1 路模拟量输出的 SB

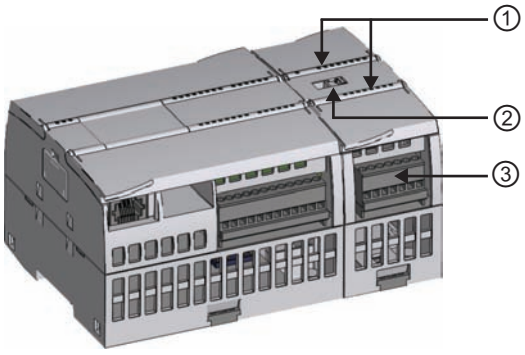


① SB 上的状态 LED

② 可拆卸用户接线连接器

1.3 信号模块

可以使用信号模块给 CPU 增加附加功能。 信号模块连接在 CPU 右侧。



① 信号模块的 I/O 的状态 LED

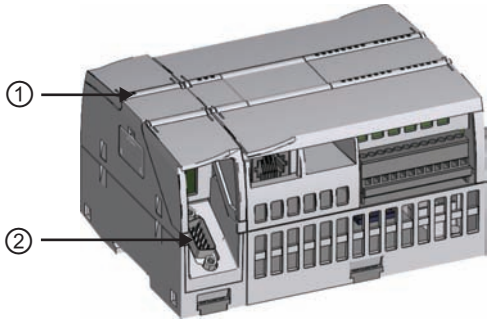
② 总线连接器

③ 可拆卸用户接线连接器

1.4 通信模块

S7-1200 系列提供了给系统增加附加功能的通信模块 (CM, Communication Module)。 有两种通信模块： RS232 和 RS485。

- CPU 最多支持 3 个通信模块
- 各 CM 连接在 CPU 的左侧（或连接到另一 CM 的左侧）



① 通信模块的状态 LED

② 通信连接器

1.5 STEP 7 Basic

STEP 7 Basic 软件提供了一个用户友好的环境，供用户开发、编辑和监视控制应用所需的逻辑，其中包括用于管理和组态项目中所有设备（例如 PLC 和 HMI 等设备）的工具。STEP 7 Basic 提供了两种编程语言（LAD 和 FBD）用于方便高效地开发适合用户具体应用的控制程序，而且还提供了用于在项目中创建和组态 HMI 设备的工具。。

为了帮助用户查找需要的信息，STEP 7 Basic 提供了内容丰富的在线帮助系统。

要安装 STEP 7 Basic，请将 CD 插入计算机的 CD-ROM 驱动器中。 安装向导自动启动并在整个安装过程中给出提示。 有关详细信息，可参考自述文件。

说明

要在运行 Windows 2000、Windows XP 或 Windows Vista 操作系统的 PC 上安装 STEP 7 Basic 软件，必须以管理员权限登录。

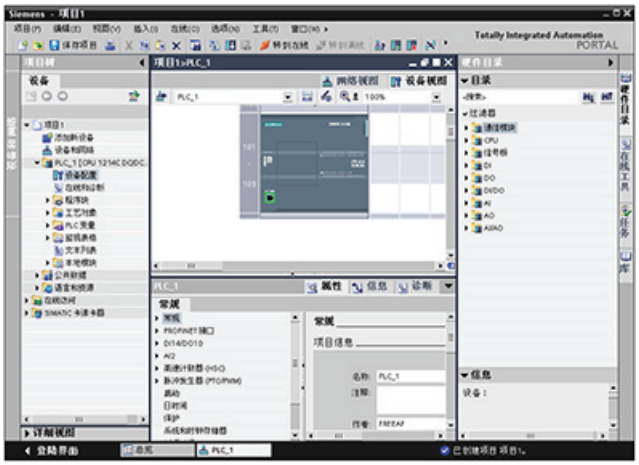
1.5.1 使工作更轻松的不同视图

为了帮助用户提高生产率，全集成自动化门户提供了两种不同的工具集视图： 根据工具功能组织的面向任务的门户集（门户视图），或项目中各元素组成的面向项目的视图（项目视图）。 请选择能让您的工作最高效的视图。 只需通过单击就可以切换门户视图和项目视图。

门户视图提供项目任务的功能视图并根据要完成的任务（例如，创建硬件组件和网络的组态）组织工具的功能。

用户可以很容易地确定如何继续以及选择哪个任务。

项目视图提供了访问项目中任意组件的途径。 由于这些组件组织在一个视图中，所以您可以方便地访问项目的各个方面。 项目包含已创建或已完成的所有元素。



1.5.2 在您需要时提供的帮助

快速查找问题答案

为了帮助用户快速高效地解决问题，STEP 7 Basic 提供了智能的需求点帮助：

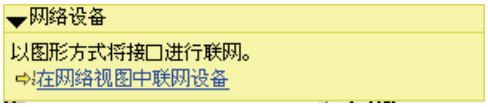
- 输入域提供“弹出式”帮助以帮助用户输入适合该域的正确信息（有效的范围和数据类型）。 例如，如果输入无效值，则将弹出一个消息文本框来提供有效值的范围。
- 界面中的某些工具提示（例如，指令的工具提示）通过“层叠”提供更多信息。 一些层叠工具提示会链接到在线信息系统（在线帮助）中的特定主题。

此外，STEP 7 Basic 还具有丰富全面的信息系统，完整介绍了 SIMATIC 工具的功能。

弹出式帮助和层叠工具提示

各种对话框和任务卡的输入域以消息框的形式提供反馈，这些消息框会弹出并给出所需的数据范围或类型。

软件界面上的元素提供工具提示来说明元素的功能。 一些元素（例如，“打开”或“保存”图标）不需要更多信息。 但有些元素提供了可显示元素附加描述的机制。 该附加信息“层叠”在来自工具提示的框中。 （工具提示旁的黑色三角形表示有更多信息。）



将光标悬停在软件界面的元素上会显示工具提示。 要显示附加信息，只需将光标悬停在工具提示上。 一些层叠工具提示还提供了指向信息系统中相关主题的链接。 单击链接将显示具体的主题。

信息系统

STEP 7 Basic 提供了丰富全面的在线信息和帮助系统，该系统介绍了用户已安装的所有 SIMATIC 产品。 该信息系统还包含参考信息和实例。 要显示该信息系统，请从以下访问点进行选择：

- 从门户视图，选择起始门户并单击“帮助”(Help) 命令。
- 从项目视图，在“帮助”(Help) 菜单中选择“显示帮助”(Show help) 命令。
- 从层叠工具提示，单击链接以显示相应主题的更多信息。

该信息系统会在一个不会遮挡工作区域的窗口中打开。

单击信息系统中的“显示/隐藏目录”按钮可显示目录和移除帮助窗口。 随后可以调整帮助窗口的大小。 使用“目录”(Contents) 或“索引”(Index) 选项卡可以按主题或关键字搜索整个信息系统。

帮助窗口（默认）	显示有目录的帮助窗口
	

说明

如果 STEP 7 Basic 已最大化，则单击“显示/隐藏目录”按钮将不会移除帮助窗口。 单击“向下恢复”按钮可移除帮助窗口。 随后可以移动和调整帮助窗口的大小。

从信息系统中打印主题

要从信息系统中打印，请单击帮助窗口中的“打印”(Print) 按钮。

要从信息系统中打印，请单击帮助窗口中的“打印”(Print) 按钮。

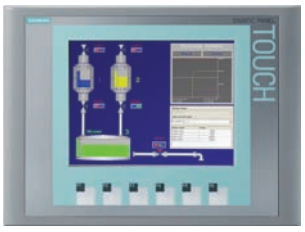
通过“打印”(Print) 对话框可以选择要打印的主题。 确保面板显示了主题。 然后可以选择任何要打印的其它主题。 单击“打印”(Print) 按钮将所选主题发送到打印机。

1.6 显示面板

由于可视化已成为大多数机器设计的标准组件，所以 SIMATIC HMI 基本型面板提供了用于执行基本操作员监控任务的触摸屏设备。所有面板的保护等级均为 IP65 并通过 CE、UL、cULus 和 NEMA 4x 认证。

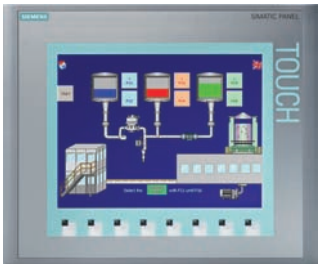
KTP 400 Basic PN

- 单色（STN，灰度）
- 4" 触摸屏，带 4 个触摸键
- 纵向或横向
- 尺寸：3.8"
- 分辨率：320 x 240
- 128 个变量
- 50 个过程画面
- 200 个报警
- 25 条曲线
- 32 KB 配方存储器
- 5 个配方，20 条数据记录，20 个条目



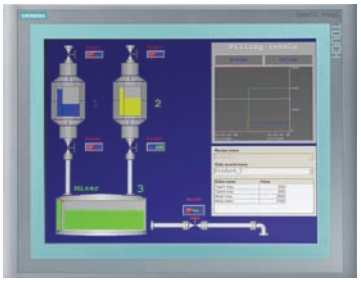
KTP 600 Basic PN

- 彩色（TFT，256 色）或单色（STN，灰度）
- 6" 触摸屏，带 6 个触摸键
- 纵向或横向
- 尺寸：5.7"
- 分辨率：320 x 240
- 128 个变量
- 50 个过程画面
- 200 个报警
- 25 条曲线
- 32 KB 配方存储器
- 5 个配方，20 条数据记录，20 个条目



KTP1000 Basic PN

- 彩色（TFT，256 色）
- 10" 触摸屏，带 8 个触摸键
- 尺寸：10.4"
- 分辨率： 640 x 480
- 256 个变量
- 50 个过程画面
- 200 个报警
- 25 条曲线
- 32 KB 配方存储器
- 5 个配方，20 条数据记录，20 个条目



TP1500 Basic PN

- 彩色（TFT，256 色）
- 15" 触摸屏
- 尺寸：15.1"
- 分辨率： 1024 x 768
- 256 个变量
- 50 个过程画面
- 200 个报警
- 25 条曲线
- 32 KB 配方存储器（集成闪存）
- 5 个配方，20 条数据记录，20 个条目

2 安装

S7-1200 设备设计得易于安装。 可以将 S7-1200 安装在面板或标准导轨上，并且可以水平或垂直安装 S7-1200。 S7-1200 尺寸较小，用户可以有效地利用空间。

⚠警告

SIMATIC S7-1200 PLC 是敞开式控制器。 需要将 S7-1200 安装在外壳、控制柜或电控室内。 仅限获得授权的人员能打开外壳、控制柜或进入电控室。

不遵守这些安装要求可能会导致死亡、人员重伤和/或财产损失。

安装 S7-1200 PLC 时务必遵守这些要求。

将 S7-1200 设备与热辐射、高压和电噪声隔离开

作为布置系统中各种设备的基本规则，必须将产生高压和高电噪声的设备与 S7-1200 等低压逻辑型设备隔离开。

在面板上配置 S7-1200 的布局时，请考虑发热设备并将电子式设备布置在控制柜中较凉爽区域。 少暴露在高温环境中会延长所有电子设备的使用寿命。

另外还要考虑面板中设备的布线。 避免将低压信号线和通信电缆铺设在具有交流动力线和高能量快速开关直流线的槽中。

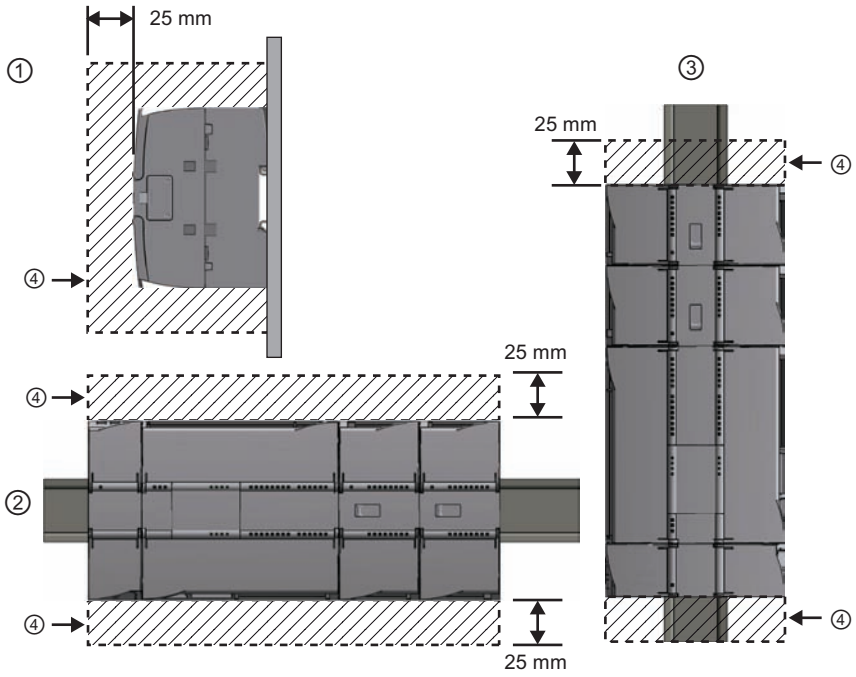
留出足够的空隙以便冷却和接线

S7-1200 被设计成通过自然对流冷却。 为保证适当冷却，在设备上方和下方必须留出至少 25 mm 的空隙。 此外，模块前端与机柜内壁间至少应留出 25 mm 的深度。

⚠小心

垂直安装时，允许的最大环境温度将降低 10 摄氏度。请调整垂直安装的 S7-1200 系统的方位使 CPU 处于低端。

规划 S7-1200 系统的布局时，应留出足够的空隙以方便接线和通信电缆连接。



① 侧视图

② 水平安装

③ 垂直安装

④ 空隙区域

功率预算

CPU 有一个内部电源，用于为 CPU、信号模块、信号板和通信模块供电以及用于满足其它 24 VDC 用户的功率要求。

有关 CPU 所提供的 5 VDC 逻辑预算和信号模块、信号板和通信模块的 5 VDC 功率要求的信息，请参考技术规范 (页 321)。 请参考“计算功率预算” (页 375)来确定 CPU 可以为您的配置提供多少电能（或电流）。

CPU 提供 24 VDC 传感器电源，该电源可以为输入点、信号模块上的继电器线圈电源或其它要求供给 24 VDC。 如果您的 24 VDC 功率要求超出该传感器电源的预算，则必须给系统增加外部 24 VDC 电源。 有关具体 S7-1200 CPU 的 24 VDC 传感器电源功率预算，请参考技术规范 (页 321)。

如果需要外部 24 VDC 电源，请确保该电源不要与 CPU 的传感器电源并联。 为提高电噪声防护能力，建议连接不同电源的公共端 (M)。

⚠警告

将外部 24 VDC 电源与 24 VDC 传感器电源并联会导致这两个电源之间有冲突，因为每个电源都试图建立自己首选的输出电压电平。

该冲突可能使其中一个电源或两个电源的寿命缩短或立即出现故障，从而导致 PLC 系统的运行不确定。 运行不确定可能导致死亡、人员重伤和/或财产损失。

DC 传感器电源和任何外部电源应分别给不同位置供电。

S7-1200 系统中的一些 24 VDC 电源输入端口是互连的，并且通过一个公共逻辑电路连接多个 M 端子。 例如，在数据表中指定为“非隔离”时，以下电路是互连的： CPU 的 24 VDC 电源、SM 的继电器线圈的电源输入或非隔离模拟输入的电源。 所有非隔离的 M 端子必须连接到同一个外部参考电位。

⚠警告

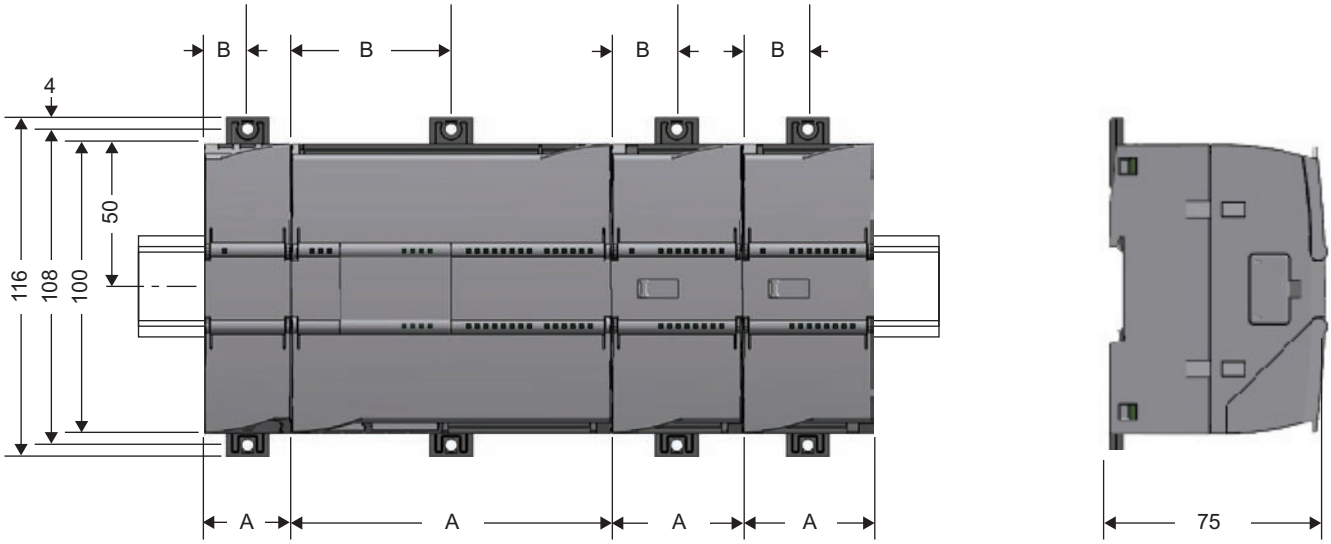
将非隔离的 M 端子连接到不同参考电位将导致意外的电流，该电流可能导致 PLC 和任何连接设备损坏或运行不确定。

不遵守这些准则可能会导致设备损坏或运行不确定，而后者可能导致死亡、人员重伤和/或财产损失。

务必确保 S7-1200 系统中的所有非隔离 M 端子都连接到同一个参考电位。

2.2 安装和拆卸步骤

安装尺寸 (mm)



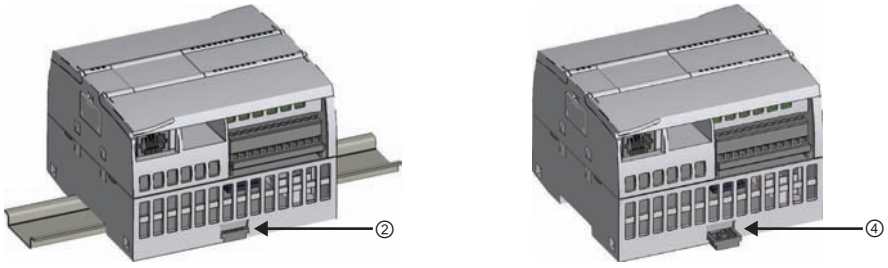
S7-1200 设备		宽度 A	宽度 B
CPU：	CPU 1211C 和 CPU 1212C	90 mm	45 mm
	CPU 1214C	110 mm	55 mm
信号模块：	8 和 16 点 DC 和继电器型（8I、16I、8Q、16Q、8I/8Q） 模拟量（4AI、8AI、4AI/4AQ、2AQ、4AQ）	45 mm	22.5 mm
	16I/16Q 继电器型 (16I/16Q)	70 mm	35 mm
通信模块：	CM 1241 RS232 和 CM 1241 RS485	30 mm	15 mm

CPU、SM 和 CM 支持 DIN 导轨安装和面板安装。 使用模块上的 DIN 导轨卡夹将设备固定到导轨上。 这些卡夹还能掰到一个伸出位置以提供将设备直接安装到面板上的螺钉安装位置。 设备上 DIN 卡夹的孔内部尺寸是 4.3 mm。

必须在设备的上方和下方留出 25 mm 的发热区以便空气自由流通。

安装和拆卸 S7-1200 设备

CPU 可以很方便地安装到标准 DIN 导轨或面板上。 可使用 DIN 导轨卡夹将设备固定到 DIN 导轨上。 这些卡夹还能掰到一个伸出位置以提供设备面板安装时所用的螺钉安装位置。



① ③

① DIN 导轨安装

② DIN 导轨卡夹处于锁紧位置

③ 面板安装

④ 卡夹处于伸出位置用于面板安装

在安装或拆卸任何电气设备之前，请确保已关闭相应设备的电源。同时，还要确保已关闭所有相关设备的电源。

⚠警告

安装或拆卸已上电的 S7-1200 或相关设备可能会导致电击或意外设备操作。

如果在安装或拆卸过程中没有断开 S7-1200 或相关设备的所有电源，则可能会由于电击或意外设备操作而导致死亡、人员重伤和/或财产损失。

务必遵守适当的安全预防措施，确保在尝试安装或拆卸 S7-1200 CPU 或相关设备前断开 S7-1200 的电源。

务必确保无论何时更换或安装 S7-1200 设备，都使用正确的模块或同等设备。

⚠警告

S7-1200 模块安装不当可能导致 S7-1200 中的程序工作异常。

如果不是用相同型号、方向或顺序来更换 S7-1200 设备，则可能会由于意外设备操作而导致死亡、人员重伤和/或财产损失。

请使用相同型号的设备来更换 S7-1200 设备，并确保设备的方向和位置放置正确。

2.2.1 安装和拆卸 CPU

安装

可以将 CPU 安装到 DIN 导轨或面板上。

说明

将全部通信模块连接到 CPU 上，然后将该组件作为一个单元来安装。 在安装 CPU 之后分别安装信号模块。

要将 CPU 安装到面板上，请按以下步骤操作：

1. 按照安装尺寸图所示的尺寸，执行定位、钻孔和攻丝以准备安装孔（M4 或美国标准 8 号）。
2. 从模块上掰出安装卡夹。 确保 CPU 上部和下部的 DIN 导轨卡夹都处于伸出位置。
3. 使用放到卡夹中的螺钉将模块固定到面板上。

说明

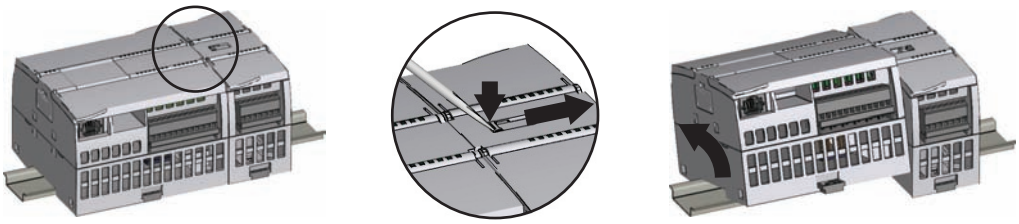
如果系统处在多振动环境或采用垂直安装，则通过面板安装 S7-1200 将能提供更高的防护等级。

要将 CPU 安装到 DIN 导轨上，请按以下步骤操作：

1. 安装 DIN 导轨。 每隔 75 mm 将导轨固定到安装板上。
2. 将 CPU 挂到 DIN 导轨上方。
3. 拉出 CPU 下方的 DIN 导轨卡夹以便能将 CPU 安装到导轨上。
4. 向下转动 CPU 使其在导轨上就位。
5. 推入卡夹将 CPU 锁定到导轨上。

拆卸

若要准备拆卸 CPU，请断开 CPU 的电源及其 I/O 连接器、接线或电缆。 将 CPU 和所有相连的通信模块作为一个完整单元拆卸。 所有信号模块应保持安装状态。



如果信号模块已连接到 CPU，则需要缩回总线连接器：

1. 将螺丝刀放到信号模块上方的小接头旁。
2. 向下按使连接器与 CPU 相分离。
3. 将小接头完全滑到右侧。

卸下 CPU：

1. 拉出 DIN 导轨卡夹从导轨上松开 CPU。
2. 向上转动 CPU 使其脱离导轨，然后从系统中卸下 CPU。

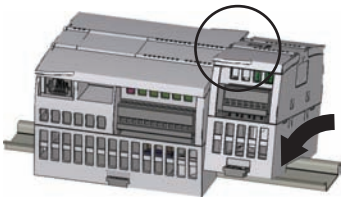
2.2.2 安装和拆卸信号模块

安装

在安装 CPU 之后安装 SM。

卸下 CPU 右侧的连接器盖。

- 将螺丝刀插入盖上方的插槽中。
- 将其上方的盖轻轻撬出并卸下盖。 收好盖以备再次使用。

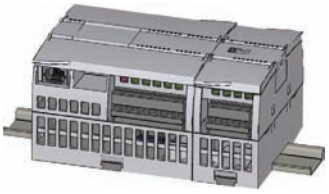


将 SM 装在 CPU 旁边。

1. 将 SM 挂到 DIN 导轨上方。
2. 拉出下方的 DIN 导轨卡夹以便将 SM 安装到导轨上。
3. 向下转动 CPU 旁的 SM 使其就位并推入下方的卡夹将 SM 锁定到导轨上。

伸出总线连接器。

1. 将螺丝刀放到 SM 上方的小接头旁。
2. 将小接头滑到最左侧，使总线连接器伸到 CPU 中。



伸出总线连接器即为 SM 建立了机械和电气连接。

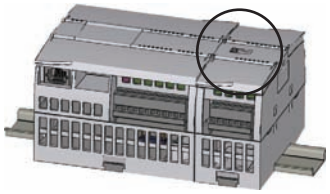
要接着信号模块再安装信号模块，请按照相同的步骤操作。

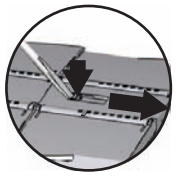
拆卸

可以在不卸下 CPU 或其它 SM 处于原位时卸下任何 SM。 若要准备拆卸 SM，请断开 CPU 的电源并卸下 SM 的 I/O 连接器和接线。

缩回总线连接器。

1. 将螺丝刀放到 SM 上方的小接头旁。
2. 向下按使连接器与 CPU 相分离。
3. 将小接头完全滑到右侧。

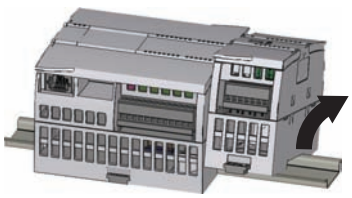




如果右侧还有 SM，则对该 SM 重复该步骤。

卸下 SM：

1. 拉出下方的 DIN 导轨卡夹从导轨上松开 SM。
2. 向上转动 SM 使其脱离导轨。 从系统中卸下 SM。
3. 如有必要，用盖子盖上 CPU 的总线连接器以避免污染。



要拆除信号模块旁的信号模块，请按照相同的步骤操作。

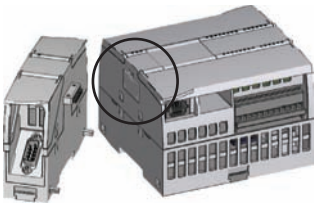
2.2.3 安装和拆卸通信模块

安装

请首先将 CM 连接到 CPU 上，然后再将整个组件作为一个单元安装到 DIN 导轨或面板上。

卸下 CPU 左侧的总线盖：

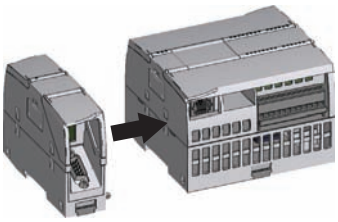
1. 将螺丝刀插入总线盖上方的插槽中。
2. 轻轻撬出上方的盖。

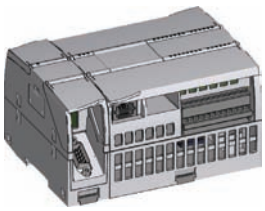


卸下总线盖。 收好盖以备再次使用。

连接单元：

1. 使 CM 的总线连接器和接线柱与 CPU 上的孔对齐。
2. 用力将两个单元压在一起直到接线柱卡入到位。





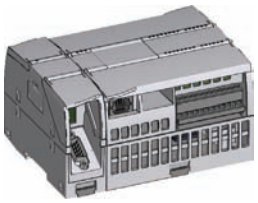
将该组合单元安装到 DIN 导轨或面板上。

1. 若是 DIN 导轨安装，确保 CPU 和相连 CM 的上部 DIN 导轨卡夹处于锁紧（内部）位置而下部 DIN 导轨卡夹处于伸出位置。
2. 如安装和拆卸 CPU (页 28) 中所示安装 CPU 与相连的 CM。
3. 将设备安装到 DIN 导轨上后，将下部 DIN 导轨卡夹推到锁紧位置以将设备锁定在 DIN 导轨上。

若是面板安装，确保将 DIN 导轨卡夹推到伸出位置。

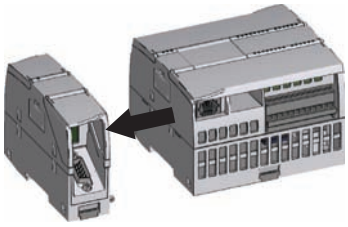
拆卸

将 CPU 和 CM 作为一个完整单元从 DIN 导轨或面板上卸下。



准备拆卸 CM。

1. 断开 CPU 的电源。
2. 拆除 CPU 和 CM 上的 I/O 连接器和所有接线及电缆。
3. 对于 DIN 导轨安装，将 CPU 和 CM 上的下部 DIN 导轨卡夹掰到伸出位置。
4. 从 DIN 导轨或面板上卸下 CPU 和 CM。



卸下 CM。

1. 用力抓住 CPU 和 CM。
2. 将它们分开。

请不要使用工具来分离这两个模块，因为这可能会损坏单元。

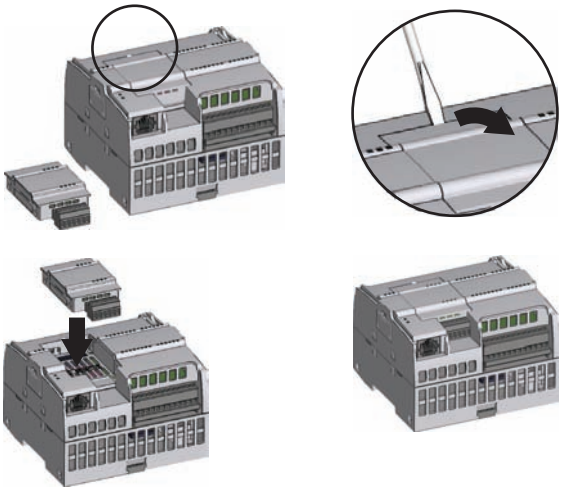
2.2.4 安装和拆卸信号板

安装

通过断开 CPU 的电源并卸下 CPU 上部和下部的端子板盖子，准备给 CPU 安装 SB。

要安装 SB，请按以下步骤操作：

1. 将螺丝刀插入 CPU 上部接线盒盖背面的槽中。
2. 轻轻将盖撬起并从 CPU 上卸下。
3. 将 SB 直接向下放入 CPU 上部的安装位置中。
4. 用力将 SB 压入该位置直到卡入就位。
5. 重新装上端子板盖子。

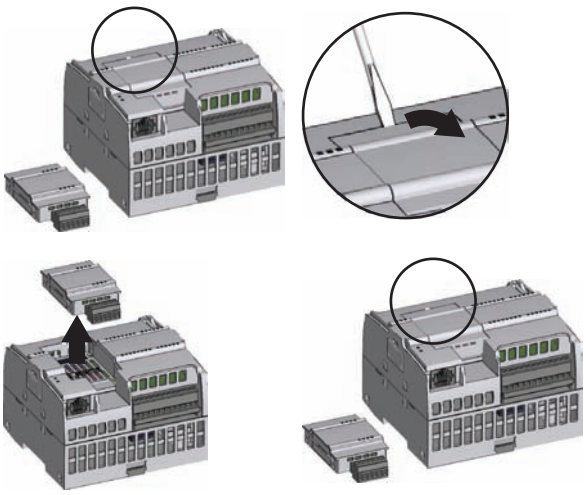


拆卸

通过断开 CPU 的电源并卸下 CPU 上部和下部的端子板盖子，准备从 CPU 上卸下 SB。

要卸下 SB，请按以下步骤操作：

1. 将螺丝刀插入 SM 上部的槽中。
2. 轻轻将 SB 撬起使其与 CPU 分离。
3. 将 SB 直接从 CPU 上部的安装位置中取出。
4. 重新装上 SB 盖。
5. 重新装上端子板盖子。



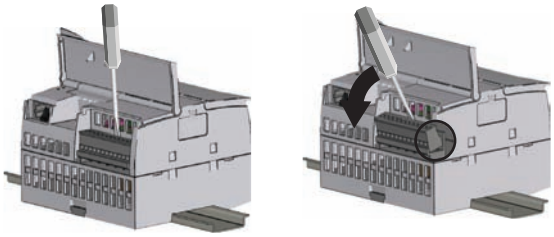
2.2.5 拆卸和重新安装 S7-1200 端子板连接器

CPU、SB 和 SM 模块提供了方便接线的可拆卸连接器。 从系统中拆卸端子板连接器的准备工作：

- 断开 CPU 的电源。
- 打开连接器上方的盖子。

要卸下连接器，请按以下步骤操作：

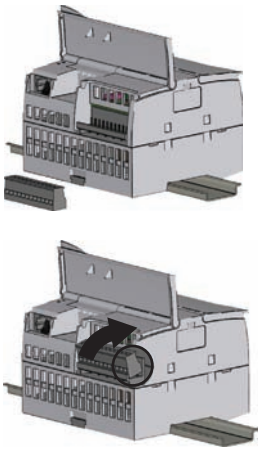
1. 查看连接器的顶部并找到可插入螺丝刀头的槽。
2. 将螺丝刀插入槽中。
3. 轻轻撬起连接器顶部使其与 CPU 分离。 连接器从夹紧位置脱离。
4. 抓住连接器并将其从 CPU 上卸下。



要安装连接器，请按以下步骤操作：

1. 通过断开 CPU 的电源并打开端子板的盖子，准备端子板安装的组件。
2. 使连接器与单元上的插针对齐。
3. 将连接器的接线边对准连接器座沿的内侧。
4. 用力按下并转动连接器直到卡入到位。

仔细检查以确保连接器已正确对齐并完全啮合。



2.3 接线准则

所有电气设备的正确接地和接线非常重要，因为这有助于确保实现最佳系统运行以及为您的应用和 S7-1200 提供更好的电噪声防护。请参考技术规范 (页 321)以查看 S7-1200 的接线图。

先决条件

在对任何电气设备进行接地或者接线之前，请确保设备的电源已经断开。同时，还要确保已关闭所有相关设备的电源。

确保在对 S7-1200 和相关设备接线时遵守所有适用的电气规程。 请根据所有适用的国家和地方标准来安装和操作所有设备。 请联系当地的管理机构确定哪些规范和标准适用于您的具体情况。

⚠警告

安装已上电的 S7-1200 或相关设备或者为这些设备接线可能会导致电击或意外设备操作。 如果在安装或拆卸过程中没有断开 S7-1200 或相关设备的所有电源，则可能会由于电击或意外设备操作而导致死亡、人员重伤和/或财产损失。

务必遵守适当的安全预防措施，确保在尝试安装或拆卸 S7-1200 或相关设备前断开 S7-1200 的电源。

在您规划 S7-1200 系统的接地和接线时，务必考虑安全问题。 电子控制设备（如 S7-1200）可能会失灵和导致正在控制或监视的设备出现意外操作。 因此，应采取一些独立于 S7-1200 的安全措施以防止可能的人员受伤或设备损坏。

> **⚠警告**
>
> 控制设备在不安全情况下运行时可能会出现故障，从而导致受控设备的意外操作。 这种意外操作可能会导致死亡、人员重伤和/或财产损失。
>
> 应使用紧急停止功能、机电超控功能或其它独立于 S7-1200 的冗余安全功能。

绝缘准则

S7-1200 交流电源和 I/O 与交流电路的边界经过设计，经验证可以在交流线路电压与低压电路之间实现安全隔离。 根据各种适用的标准，这些边界包括双重或加强绝缘，或者基本绝缘加辅助绝缘。 跨过这些边界的组件（例如，光耦合器、电容器、变压器和继电器）已通过安全隔离认证。 满足这些要求的绝缘边界在 S7-1200 产品数据页中被标识为具有 1500 VAC 或更高的绝缘度。 该标识是通过准许的方法采用 (2Ue + 1000 VAC) 或等效电压进行常规工厂测试得来的。 S7-1200 的安全隔离边界已通过高达 4242 VDC 的典型试验。

根据 EN 61131-2，集成有交流电源的 S7-1200 的传感器电源输出、通信电路和内部逻辑电路属于 SELV（安全超低电压）电路。

要维持 S7-1200 低压电路的安全特性，到通信端口、模拟电路以及所有 24 V 额定电源和 I/O 电路的外部连接必须由合格的电源供电，该电源必须满足各种标准对 SELV、PELV、二类、限制电压或受限电源的要求。

> **⚠警告**
>
> 若使用非隔离或单绝缘电源通过交流线路给低压电路供电，可能会导致本来应当可以安全触摸的电路上出现危险电压，例如，通信电路和低压传感器线路。
>
> 这种意外的高压可能会引起电击而导致死亡、人员重伤和/或财产损失。
>
> 只应当使用合格的高压转低压整流器作为可安全接触的限压电路的供电电源。

S7-1200 的接地准则

将应用设备接地的最佳方式是确保 S7-1200 和相关设备的所有公共端和接地连接在同一个点接地。该点应该直接连接到系统的大地接地。

所有地线应尽可能地短且应使用大线径，例如，2 mm^2 (14 AWG)。

确定接地点时，应考虑安全接地要求和保护性中断装置的正常运行。

S7-1200 的接线准则

规划 S7-1200 的接线时，应提供一个可同时切断 S7-1200 CPU 电源、所有输入电路和所有输出电路电力供应的隔离开关。 请提供过流保护（例如，熔断器或断路器）以限制电源线中的故障电流。 考虑在各输出电路中安装熔断器或其它电流限制器提供额外保护。

为所有可能遭雷电冲击的线路安装合适的浪涌抑制设备。

避免将低压信号线和通信电缆铺设在具有交流线和高能量快速开关直流线的槽中。始终成对布线，中性线或公共线与火线或信号线成对。

使用尽可能短的电线并确保线径适合承载所需电流。 连接器接受 2 mm^2 到 0.3 mm^2（14 AWG 到 22 AWG）的线径。使用屏蔽线以便最好地防止电噪声。 通常在 S7-1200 端将屏蔽层接地能获得最佳效果。

在给通过外部电源供电的输入电路接线时，应在电路中安装过流保护装置。 由 S7-1200 的 24 VDC 传感器电源供电的电路不需要外部保护，因为该传感器电源的电流已经受到限制。

所有 S7-1200 模块都有供用户接线的可拆卸连接器。要防止连接器松动，请确保连接器固定牢靠并且导线被牢固地安装到连接器中。为避免损坏连接器，小心不要将螺丝拧得过紧。 连接器螺钉的最大扭矩为 0.56 N-m（5 英寸-磅）。

为了有利于防止安装中出现意外的电流，S7-1200 在某些点提供绝缘边界。 在您规划系统的接线时，应考虑这些绝缘边界。 有关所提供的绝缘程度和绝缘边界位置的信息，请参见技术规范。 不要相信额定值小于 1500 VAC 的绝缘边界是安全边界。

感性负载的使用准则

应当为感性负载安装抑制电路，限制在关闭控制输出时的电压上升。 抑制电路可保护输出，防止关闭感性负载时产生的高压导致其过早损坏。 此外，抑制电路还能限制开关感性负载时产生的电噪声。 布置一个外部抑制电路使其从电路上跨接在负载两端并且在位置上接近负载，这样对降低电气噪声最有效。

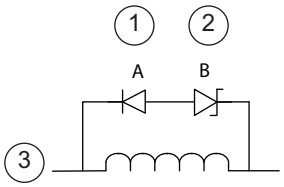
说明

给定的抑制电路是否有效取决于实际的应用，必须针对具体应用检验其有效性。 务必确保抑制电路中使用的所有元件都适合您的具体应用。

控制直流感性负载

S7-1200 的 DC 输出包括抑制电路，该电路足以抑制大多数应用的感性负载。 由于继电器可用于直流或交流负载，所以未提供内部保护。 下图显示了一个直流负载抑制电路实例。

在大多数应用中，在感性负载两端增加一个二极管 (A) 就可以了，但如果您的应用要求更快的关闭时间，则建议再增加一个稳压二极管 (B)。



① I1N4001 二极管或同等元件

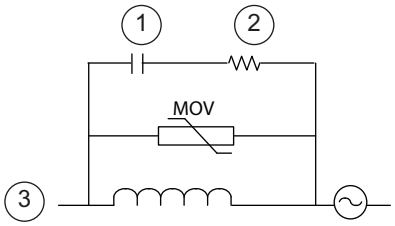
② 8.2 V 稳压二极管（直流输出），
36 V 稳压二极管（继电器输出）

③ 输出点

请确保正确选择稳压二极管，以适合输出电路中的电流量。

控制交流负载的继电器输出

使用继电器输出开关 115 V/230 VAC 负载时，请在交流负载两端并联一个电阻/电容网络，如图所示。 也可以使用金属氧化物变阻器 (MOV) 限制尖峰电压。 请确保 MOV 的工作电压至少比额定线电压高出 20%。



① 0.1 µ F

② 100 到 120 Ω

③ 输出点

灯负载的使用准则

由于接通浪涌电流大，灯负载会损坏继电器触点。 该浪涌电流通常是钨灯稳态电流的 10 到 15 倍。 对于在应用期间将进行大量开关操作的灯负载，建议安装可更换的插入式继电器或浪涌限制器。

3 PLC 概念

3.1 用户程序的执行

CPU 支持以下类型的代码块，使用它们可以创建有效的用户程序结构：

- 组织块 (OB) 定义程序的结构。 有些 OB 具有预定义的行为和启动事件，但用户也可以创建具有自定义启动事件的 OB。
- 功能 (FC) 和功能块 (FB) 包含与特定任务或参数组合相对应的程序代码。 每个 FC 或 FB 都提供一组输入和输出参数，用于与调用块共享数据。 FB 还使用相关联的数据块（称为背景数据块）来保存执行期间的值状态，程序中的其它块可以使用这些值状态。
- 数据块 (DB) 存储程序块可以使用的数据。

用户程序的执行顺序是：从一个或多个在进入 RUN 模式时运行一次的可选启动组织块 (OB) 开始，然后执行一个或多个循环执行的程序循环 OB。 OB 也可以与中断事件（可以是标准事件或错误事件）相关联，并在相应的标准或错误事件发生时执行。

功能 (FC) 或功能块 (FB) 是指可从 OB 或其它 FC/FB 调用的程序代码块，可下至以下层级：

- 16（从程序循环 OB 或启动 OB 开始）
- 4（从延时中断、循环中断、硬件中断、时间错误中断或诊断错误中断 OB 开始）

FC 不与任何特定数据块 (DB) 相关联，而 FB 与 DB 直接相关并使用 DB 来传送参数以及存储中间值和结果。

用户程序、数据及组态的大小受 CPU 中可用装载存储器和工作存储器的限制。 在可用工作存储器空间范围内，对所支持的块数量没有限制。

每个周期都包括写入输出、读取输入、执行用户程序指令以及执行系统维护或后台处理。该周期称为扫描周期或扫描。

只有在通电时，才会对信号板、信号模块和通信模块进行检测和注册。

说明

不支持在通电时（热）插入和拔出信号板、信号模块和通信模块。 唯一的例外是 SIMATIC 存储卡，它可以在 CPU 通电时插入或拔出。

在默认组态中，所有数字量和模拟量 I/O 点都通过内部存储区（即过程映像）与扫描周期同步更新。 过程映像包含物理输入和输出（CPU、信号板和信号模块上的物理 I/O 点）的快照。

CPU 执行以下任务：

- CPU 将过程映像输出区中的输出值写入到物理输出。
- CPU 仅在用户程序执行前读取物理输入，并将输入值存储在过程映像输入区。 这样可确保这些值在整个用户指令执行过程中保持一致。
- CPU 执行用户指令逻辑，并更新过程映像输出区中的输出值，而不是写入实际的物理输出。

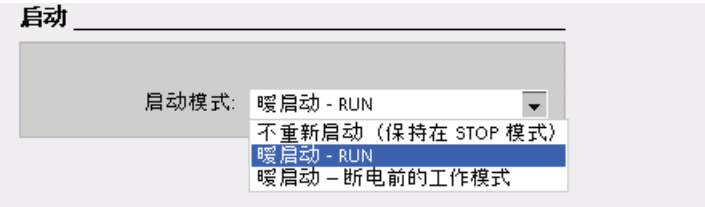
这一过程通过在给定周期内执行用户指令而提供一致的逻辑，并防止物理输出点可能在过程映像输出区中多次改变状态而出现抖动。

用户可以指定是否将数字量和模拟量 I/O 点存储到过程映像中。 如果在设备视图中插入模块，则其数据将默认存储到 S7-1200-CPU 的过程映像中。 CPU 在过程映像更新期间自动处理模块和过程映像间的数据交换。 要从过程映像自动更新中删除数字量或模拟量点，请在设备配置中选择相应的设备，查看“属性”(Properties) 选项卡，在必要时展开以查找所需 I/O 点，然后选择“IO 地址/硬件标识符”(IO addresses/HW identifier)。 然后将“过程映像：”(Process image:) 对应的条目从“循环 PI”(Cyclic PI) 更改为“---”。 要将这些点重新添加到过程映像自动更新中，请将该选项再更改为“循环 PI”(Cyclic PI)。

可以在指令执行时立即读取物理输入值和立即写入物理输出值。 无论 I/O 点是否被组态为存储到过程映像中，立即读取功能都将访问物理输入的当前状态而不更新过程映像输入区。 立即写入物理输出功能将同时更新过程映像输出区（如果相应 I/O 点组态为存储到过程映像中）和物理输出点。 如果想要程序不使用过程映像，直接从物理点立即访问 I/O 数据，则在 I/O 地址后加后缀“:P”。

组态启动参数

使用 CPU 属性可组态 CPU 在通电周期后的启动方式。



选择 CPU 是在 STOP 模式、RUN 模式还是上一个模式（通电周期之前）下启动。

CPU 在进入 RUN 模式前执行暖启动。 暖启动会将所有非保持性存储器复位为默认初始值，但保留保持性存储器中存储的当前值。

说明

下载完成后 CPU 总是会执行重新启动

每次下载完项目元素（例如程序块、数据块或硬件配置），CPU 都会在下一次切换到 RUN 模式时先执行重新启动。 除清除输入、初始化输出以及初始化非保持性存储器之外，重新启动还会初始化保持性存储区。

在紧随下载的重新启动完成之后，所有随后的 STOP 到 RUN 切换均会执行暖启动（不会初始化保持性存储器）。

3.1.1 CPU 的工作模式

CPU 有以下三种工作模式： STOP 模式、STARTUP 模式和 RUN 模式。 CPU 前面的状态 LED 指示当前工作模式。

- 在 STOP 模式下，CPU 不执行任何程序，而用户可以下载项目。
- 在 STARTUP 模式下，执行一次启动 OB（如果存在）。 在 RUN 模式的启动阶段，不处理任何中断事件。
- 在 RUN 模式下，重复执行扫描周期。 中断事件可能会在程序循环阶段的任何点发生并进行处理。

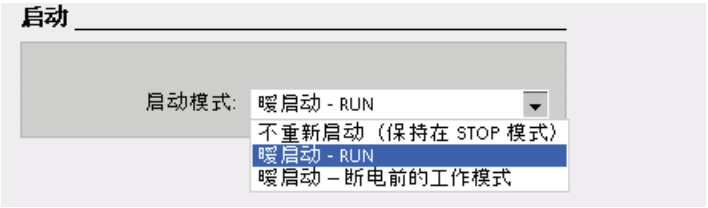
  处于 RUN 模式下时，无法下载任何项目。

CPU 支持通过暖启动进入 RUN 模式。 暖启动不包括储存器复位。 在暖启动时，所有非保持性系统及用户数据都将被初始化。 保留保持性用户数据。

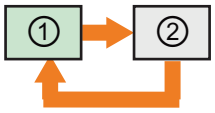
存储器复位将清除所有工作存储器、保持性及非保持性存储区，并将装载存储器复制到工作存储器。 存储器复位不会清除诊断缓冲区，也不会清除永久保存的 IP 地址值。

可以使用编程软件指定 CPU 的上电模式以及重启方法。 该组态项目出现在 CPU“设备配置”(Device Configuration) 的“启动”(Startup) 下。 通电后，CPU 将执行一系列上电诊断检查和系统初始化操作。 然后 CPU 进入适当的上电模式。 检测到的某些错误将阻止 CPU 进入 RUN 模式。 CPU 支持以下上电模式：

- STOP 模式
- 暖启动后转到 RUN 模式
- 暖启动后转到上一个模式



使用编程软件在线工具中的“STOP”或“RUN”命令，可以更改当前工作模式。 也可在程序中包含 STP 指令，以使 CPU 切换到 STOP 模式。 这样就可以根据程序逻辑停止程序的执行。

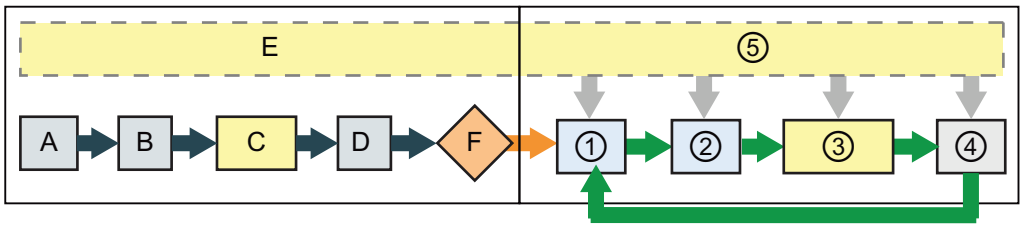


在 STOP 模式下，CPU ① 处理所有通信请求（如果适用）并 ② 执行自诊断。

在 STOP 模式下，CPU 不执行用户程序，过程映像也不会自动更新。

只有在 CPU 处于 STOP 模式时，才能下载项目。

在 RUN 模式下，CPU 执行下图所示的任务。



STARTUP

- A 清除 I 存储区
- B 使用上一个值或替换值对输出执行初始化
- C 执行启动 OB
- D 将物理输入的状态复制到 I 存储器
- E 将所有中断事件存储到要在 RUN 模式下处理的队列中
- F 启用 Q 存储器到物理输出的写入操作

RUN

- ① 将 Q 存储器写入物理输出
- ② 将物理输入的状态复制到 I 存储器
- ③ 执行程序循环 OB
- ④ 执行自检诊断
- ⑤ 在扫描周期的任何阶段处理中断和通信

STARTUP 过程

只要工作状态从 STOP 切换到 RUN，CPU 就会清除过程映像输入、初始化过程映像输出并处理启动 OB。 启动 OB 中的指令对过程映像输入进行任何读访问时，读取到都只有零，而不是当前物理输入值。 因此，要在启动模式下读取物理输入的当前状态，必须执行立即读取操作。 接着再执行启动 OB 以及任何相关的 FC 和 FB。 如果存在多个启动 OB，则按照 OB 编号依次执行各启动 OB，OB 编号最小的先执行。

每个启动 OB 都包含帮助您确定保持性数据和日时钟有效性的启动信息。 可以在启动 OB 中编写指令，以检查这些启动值，从而采取适当的措施。 启动 OB 支持以下启动位置：

输入	数据类型	说明
LostRetentive	BOOL	如果保持性数据存储区丢失，该位为真
LostRTC	BOOL	如果日时钟（实时时钟）丢失，该位为真

在启动过程中，CPU 还会执行以下任务。

- 在启动阶段，对中断进行排队但不加以处理
- 在启动阶段，不执行任何循环时间监视
- 在启动模式下，可以更改 HSC（High-Speed Counter，高速计数器）、PWM（Pulse-Width Modulation，脉冲宽度调制）以及 PtP（Point-to-Point communication，点对点通信）模块的组态
- 只有在 RUN 模式下才会真正运行 HSC、PWM 和点对点通信模块

执行完启动 OB 后，CPU 将进入 RUN 模式并在连续的扫描周期内处理控制任务。

在 RUN 模式下处理扫描周期

在每个扫描周期中，CPU 都会写入输出、读取输入、执行用户程序、更新通信模块、执行内部处理工作以及响应用户中断事件和通信请求。 在扫描期间会定期处理通信请求。

以上操作（用户中断事件除外）按先后顺序定期进行处理。 对于已启用的用户中断事件，则根据优先级按其发生顺序进行处理。

系统要保证扫描周期在一定的时间段内（即最大循环时间）完成；否则将生成时间错误事件。

- 在每个扫描周期的开始，从过程映像重新获取数字量及模拟量输出的当前值，然后将其写入到 CPU、SB 和 SM 模块上组态为自动 I/O 更新（默认组态）的物理输出。 通过指令访问物理输出时，输出过程映像和物理输出本身都将被更新。

- 随后在该扫描周期中，将读取 CPU、SB 和 SM 模块上组态为自动 I/O 更新（默认组态）的数字量及模拟量输入的当前值，然后将这些值写入过程映像。 通过指令访问物理输入时，指令将访问物理输入的值，但输入过程映像不会更新。
- 读取输入后，系统将从第一条指令开始执行用户程序，一直执行到最后一条指令。 其中包括所有的程序循环 OB 及其所有关联的 FC 和 FB。 程序循环 OB 根据 OB 编号依次执行，OB 编号最小的先执行。

在扫描期间会定期处理通信请求，这可能会中断用户程序的执行。

自诊断检查包括定期检查系统和检查 I/O 模块的状态。

中断可能发生在扫描周期的任何阶段，并且由事件驱动。 事件发生时，CPU 将中断扫描循环，并调用被组态用于处理该事件的 OB。 OB 处理完该事件后，CPU 从中断点继续执行用户程序。

组织块 (OB)

OB 控制用户程序的执行。 每个 OB 的 OB 编号必须唯一。 200 以下的一些默认 OB 编号被保留。其它 OB 编号必须大于或等于 200。

CPU 中的特定事件将触发组织块的执行。 OB 无法互相调用或通过 FC 或 FB 调用。 只有启动事件（例如，诊断中断或时间间隔）可以启动 OB 的执行。CPU 按优先等级处理 OB，即先执行优先级较高的 OB 然后执行优先级较低的 OB。 最低优先等级为 1（对应主程序循环），最高优先等级为 27（对应时间错误中断）。

OB 控制以下操作：

- 程序循环 OB 在 CPU 处于 RUN 模式时循环执行。 主程序块是程序循环 OB。用户在其中放置控制程序的指令以及调用其它用户块。 允许使用多个程序循环 OB，它们按编号顺序执行。 OB 1 是默认循环 OB。 其它程序循环 OB 必须标识为 OB 200 或更大。
- 启动 OB 在 CPU 的工作模式从 STOP 切换到 RUN 时执行一次，包括处于 RUN 模式时和执行 STOP 到 RUN 切换命令时上电。 之后将开始执行主“程序循环”OB。 允许有多个启动 OB。 OB 100 是默认启动 OB。 其它启动 OB 必须是 OB 200 或更大。
- 通过启动中断 (SRT_DINT) 指令组态事件后，时间延迟 OB 将以指定的时间间隔执行。 延迟时间在扩展指令 SRT_DINT 的输入参数中指定。 指定的延迟时间结束时，时间延迟 OB 将中断正常的循环程序执行。 对任何给定的时间最多可以组态 4 个时间延迟事件，每个组态的时间延迟事件只允许对应一个 OB。 时间延迟 OB 必须是 OB 200 或更大。

- 循环中断 OB 以指定的时间间隔执行。 循环中断 OB 将按用户定义的时间间隔（例如，每隔 2 秒）中断循环程序执行。 最多可以组态 4 个循环中断事件，每个组态的循环中断事件只允许对应一个 OB。 该 OB 必须是 OB 200 或更大。
- 硬件中断 OB 在发生相关硬件事件时执行，包括内置数字输入端的上升沿和下降沿事件以及 HSC 事件。 硬件中断 OB 将中断正常的循环程序执行来响应硬件事件信号。 可以在硬件配置的属性中定义事件。 每个组态的硬件事件只允许对应一个 OB。 该 OB 必须是 OB 200 或更大。
- 时间错误中断 OB 在检测到时间错误时执行。 如果超出最大循环时间，时间错误中断 OB 将中断正常的循环程序执行。 最大循环时间在 PLC 的属性中定义。 OB 80 是唯一支持时间错误事件的 OB。 可以组态没有 OB 80 时的动作： 忽略错误或切换到 STOP 模式。
- 诊断错误中断 OB 在检测到和报告诊断错误时执行。 如果具有诊断功能的模块发现错误（如果模块已启用诊断错误中断），诊断 OB 将中断正常的循环程序执行。 OB 82 是唯一支持诊断错误事件的 OB。 如果程序中没有诊断 OB，则可以组态 CPU 使其忽略错误或切换到 STOP 模式。

3.1.2 事件执行的优先级与排队

CPU 处理操作受事件控制。 由事件触发中断 OB 的执行。 事件对应的中断 OB 在创建块期间、设备配置期间或者使用 ATTACH 或 DETACH 指令指定。 有些事件定期发生，例如，程序循环或循环事件。 而其它事件只发生一次，例如，启动事件和延时事件。 有些事件在出现硬件触发的变化时发生，例如，输入点上的沿事件或高速计数器事件。 还有些事件只有在出现错误时才发生，例如，诊断错误和时间错误事件。 事件优先级、优先级组和队列用于确定事件中断 OB 的处理顺序。

程序循环事件在每个程序循环（扫描）期间发生一次 在程序循环期间，CPU 写入输出、读取输入和执行程序循环 OB。 程序循环事件是必需的，并且一直启用。 您可以不为程序循环事件选择程序循环 OB，也可选择多个 OB。 程序循环事件触发后，将执行编号最小的程序循环 OB（通常是 OB1）。 在程序循环中，其它程序循环 OB 按编号顺序依次执行。

用户可通过循环中断事件组态中断 OB 以指定的时间间隔执行。 时间间隔在创建 OB 并将其选为循环中断 OB 时组态。循环事件此后可中断程序循环并执行循环中断 OB（循环事件的优先级比程序循环事件的优先级高）。 只能将一个循环中断 OB 连接到一个循环事件。 CPU 支持 4 个循环中断事件。 循环中断 OB 具有相移属性，从而时间间隔相同的循环中断彼此错开一定的相移量执行。

启动事件在从 STOP 切换到 RUN 模式时发生一次，并触发启动 OB 执行。 可以为启动事件选择多个 OB。 启动 OB 按编号顺序执行。

用户可以通过延时中断事件组态中断 OB 在指定的延迟时间过后执行。 延迟时间使用 SRT_DINT 指令指定。 延时事件将中断程序循环以执行延时中断 OB。只能将一个延时中断 OB 连接到一个延时事件。 CPU 支持 4 个延时事件。

硬件中断事件在硬件有变化时触发，例如，输入点上的上升沿/下降沿事件或者 HSC（High Speed Counter，高速计数器）事件。 可以为每个硬件中断事件选择一个中断 OB。 硬件事件在设备配置中启用。 在硬件配置中或在用户程序中使用 ATTACH 指令为事件指定 OB。 CPU 支持多个硬件中断事件。 具体事件数取决于 CPU 型号和输入点数。

时间和诊断错误中断事件在 CPU 检测到错误时触发。 这些事件的优先级比其它中断事件的优先级高，因此可以中断延时、循环和硬件中断事件的执行。 对一个时间错误和诊断错误中断事件只能指定一个中断 OB。

了解事件执行的优先级与排队

单一来源的未决（排队的）事件数量通过各种事件类型的不同队列加以限制。 达到给定事件类型的未决事件限值后，下一个事件将丢失。 有关队列溢出的更多信息，请参见后面的“了解时间错误事件”部分。

每个 CPU 事件都有一个关联的优先级，而事件优先级分为若干个优先级组。 下表汇总了受支持 CPU 事件的队列深度、优先级组及优先级。

说明

不能更改优先级或优先级组的分配，也不能更改队列深度。

通常，事件按优先级顺序进行处理（优先级最高的最先进行处理）。 优先级相同的事件按“先到先得”的原则进行处理。

事件类型 (OB)	数量	有效 OB 编号	队列深度	优先级组	优先级
程序循环	1 个程序循环事件 允许多个 OB	1（默认） 200 或更大	1	1	1
启动	1 个启动事件 [1] 允许多个 OB	100（默认） 200 或更大	1		1
延时	4 个延时事件 每个事件 1 个 OB	200 或更大	8	2	3
循环	4 个循环事件 每个事件 1 个 OB	200 或更大	8		4
沿	16 个上升沿事件 16 个下降沿事件 每个事件 1 个 OB	200 或更大	32		5
HSC	6 个 CV = PV 事件 6 个方向更改事件 6 个外部复位事件 每个事件 1 个 OB	200 或更大	16		6
诊断错误	1 个事件	仅限 82	8		9
时间错误事件 /MaxCycle 时间事件	1 个时间错误事件 1 个 MaxCycle 时间事件	仅限 80	8	3	26
2xMaxCycle 时间事件	1 个 2xMaxCycle 时间事件	不调用 OB	-	3	27

[1] 启动事件的特殊情况

- 启动事件和程序循环事件永远不会同时发生，因为在启动事件运行完成之后才会启动程序循环事件（由操作系统控制）。
- 没有什么事件可以中断启动事件。 启动事件期间发生的事件因此将排队等到启动事件完成后再进行处理。

OB 开始执行后，如果发生另一个相同或较低优先级组中的事件，则该 OB 的处理无法被中断。 这类事件将排队等待稍后处理，从而使当前 OB 能够完成。

但是，较高优先级组中的事件可中断当前 OB，而 CPU 随后将执行较高优先级事件对应的 OB。 较高优先级 OB 完成后，CPU 将根据较高优先级组内的优先级，执行该组中排队的任何其它事件的 OB。 如果该较高优先级组中没有其它未决（排队的）事件，CPU 将返回到较低优先级组，并从被预占 OB 处理的中断点继续处理该 OB。

中断等待时间

如果中断事件发生时程序循环 OB 是唯一激活的事件服务例程，则中断事件等待时间（该时间是指从通知 CPU 发生了事件到 CPU 开始执行处理该事件的 OB 中的第一条指令）约为 210 µs。

了解时间错误事件

出现几种不同时间错误情况中的任何一种都会引起时间错误事件。 所支持的时间错误有以下几种：

- 超出最大循环时间
- 请求的 OB 无法启动
- 发生队列溢出

如果程序循环在指定的最大扫描周期时间内未完成，就会出现超出最大循环时间这种情况。 有关最大循环时间情况、如何组态最大扫描周期时间以及如何复位循环定时器的更多信息，请参见“监视循环时间 (页 45)”部分。

如果循环中断或延时中断请求 OB，但请求的 OB 已经在执行，就会出现请求的 OB 无法启动这种情况。

如果中断的出现频率超过其处理频率，就会出现发生队列溢出这种情况。 各种事件类型的未决（排队的）事件数量通过不同的队列加以限制。 如果某个事件在相应的队列已满时发生，将生成时间错误事件。

所有时间错误事件都可触发 OB 80（如果存在）的执行。 如果 OB 80 不存在，CPU 将忽略该错误。 如果在同一程序循环中出现两次超出最大循环时间的情况且没有复位循环定时器，则无论 OB 80 是否存在，CPU 都将切换到 STOP 模式。 请参见“监视循环时间 (页 45)”部分。

OB 80 中包含的启动信息有助于您确定生成时间错误的事件和 OB。 可以在 OB 80 中编写指令，以检查这些启动值以及采取适当的措施。 OB 80 支持以下启动位置：

输入	数据类型	说明
fault_id	BYTE	16#01 - 超出最大循环时间 16#02 - 请求的 OB 无法启动 16#07 和 16#09 - 发生队列溢出
csg_OBnr	OB_ANY	出错时正在执行的 OB 的编号
csg_prio	UINT	导致错误的 OB 的优先级

创建新项目时，不存在时间错误中断 OB 80。 如果需要，请在树中的“程序块”(Program blocks) 下双击“添加新块”(Add new block)，然后依次选择“组织块”(Organization block)、“时间错误中断”(Time error interrupt)，这样便可将时间错误中断 OB 80 添加到项目中。

了解诊断错误事件

某些设备能够检测和报告诊断错误。 发生或清除几种不同诊断错误情况中的任何一种都会引起诊断错误事件。 所支持的诊断错误有以下几种：

- 无用户电源
- 超出上限
- 超出下限
- 断路
- 短路

所有诊断错误事件都可触发 OB 82（如果存在）的执行。 如果 OB 82 不存在，CPU 将忽略该错误。 创建新项目时，不存在诊断错误中断 OB 82。 如果需要，请在树中的“程序块”(Program blocks) 下双击“添加新块”(Add new block)，然后依次选择“组织块”(Organization block)、“诊断错误中断”(Diagnostic error interrupt)，这样便可将诊断错误中断 OB 82 添加到项目中。

OB 82 中包含的启动信息有助于您确定是因为错误的出现还是清除导致事件的发生，以及确定报告错误的设备和通道。 可以在 OB 82 中编写指令，以检查这些启动值以及采取适当的措施。 OB 82 支持以下启动位置：

输入	数据类型	说明
IOstate	WORD	设备的 IO 状态
laddr	HW_ANY	报告错误的设备或功能单元的硬件标识符
channel	UINT	通道号
multierror	BOOL	如果存在多个错误，参数值为 TRUE（*以前版本中不支持*）

IO_state 的位 4 指示事件的发生是因为错误的出现还是清除。 错误（例如，断线）出现时位 4 为 1，错误消失后为 0。

梯形图输入包含返回错误的设备或功能单元的硬件标识符 (HW ID)。 HW ID 是在设备或网络视图中插入组件时自动分配的，它出现在 PLC 变量的“常量”(Constants) 选项卡中。还会为 HW ID 自动分配名称。PLC 变量的“常量”(Constants) 选项卡中的这些条目无法更改。

输入通道号从 0（对应第一个模拟量或数字量输入点）开始，而输出通道号从 64（对应第一个模拟量或数字量输出点）开始。 如果设备同时包含输入和输出，则需要使用不同的偏移量来区分它们。 如果错误影响了整个设备或功能单元，例如，缺少用户电源，则置位通道号（通道号 32768）字的最高有效位。

监视循环时间

循环时间是指 CPU 操作系统在 RUN 模式下执行循环阶段所需的时间。 CPU 提供了两种监视循环时间的方法：

- 最大扫描周期时间
- 固定最小扫描周期时间

扫描周期监视在启动事件完成后开始。 此功能的组态出现在 CPU“设备配置”(Device Configuration) 的“循环时间”(Cycle time) 下。

CPU 始终监视扫描周期，并在超出最大扫描周期时间时做出响应。 如果超出组态的最大扫描周期时间，将生成错误，并按以下两种方法之一对该错误进行处理：

- 如果不存在时间错误中断 OB 80，则 CPU 生成错误并继续执行用户程序
- 如果存在时间错误中断 OB 80，则 CPU 将执行 OB 80

RE_TRIGR 指令（重新触发循环时间监视）可用于复位记录循环时间的定时器。 然而，该指令只有在程序循环 OB 中执行时才起作用；在 OB 80 中执行时，RE_TRIGR 指令将被忽略。如果在同一程序循环中两次超出最大扫描周期时间，且两次之间未执行

RE_TRIGR 指令，则 CPU 将立即切换到 STOP 模式。 如果反复执行 RE_TRIGR 指令，可能会导致死循环或扫描时间非常长。

通常，扫描周期会尽快执行，当前扫描周期一完成，下一个扫描周期就会开始。 视用户程序和通信任务而定，扫描周期的时间段在各次扫描中有所不同。 为了消除这种差异，CPU 支持一种可选的固定最小扫描周期时间（也称为固定扫描周期）。 如果启用了此可选功能并且固定最小扫描周期时间的单位为 ms，则 CPU 将使完成每次 CPU 扫描的最小循环时间保持在 ±1 ms 的范围内。

如果 CPU 完成正常扫描周期的时间小于指定的最小循环时间，则 CPU 将用额外的扫描周期时间执行运行诊断和/或处理通信请求。 这样，CPU 将始终花费固定的时间量来完成扫描周期。

如果 CPU 在指定的最小循环时间内未完成扫描周期，CPU 将正常完成扫描（包括通信处理），并且不会因超出最小扫描时间而引起任何系统响应。 下表定义了循环时间监视功能的值范围和默认值。

循环时间	值范围 (ms)	默认值
最大扫描周期时间 [1]	1 到 6000	150 ms
固定最小扫描周期时间 [2]	1 到最大扫描周期时间	禁用

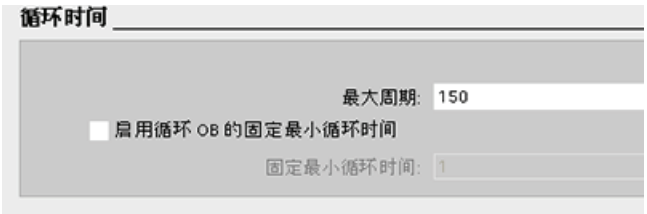
1 最大扫描周期时间始终启用。 请组态一个 1 ms 到 6000 ms 之间的周期时间。默认值为 150 ms。

2 固定最小扫描周期时间是可选的，默认情况下被禁用。 必要时，可组态一个 1 ms 到 6000 ms 之间的周期时间。

组态循环时间和通信负载

利用设备配置中的 CPU 属性可以组态以下参数：

- 循环时间 (Cycle time)： 可以输入最大扫描周期时间。 也可以输入固定最小扫描周期时间。



- 通信负载 (Communications load)： 可以组态一个百分比时间，专门用于通信任务。

有关扫描周期的更多信息，请参见监视循环时间 (页 45)。

3.1.3 CPU 存储器

存储器管理

CPU 提供了以下用于存储用户程序、数据和组态的存储区：

- 装载存储器，用于非易失性地存储用户程序、数据和组态。 项目被下载到 CPU 后，首先存储在装载存储区中。 该存储区位于存储卡（如存在）或 CPU 中。 该非易失性存储区能够在断电后继续保持。 存储卡支持的存储空间比 CPU 内置的存储空间更大。
- 工作存储器是易失性存储器，用于在执行用户程序时存储用户项目的某些内容。 CPU 会将一些项目内容从装载存储器复制到工作存储器中。 该易失性存储区将在断电后丢失，而在恢复供电时由 CPU 恢复。
- 保持性存储器，用于非易失性地存储限量的工作存储器值。 保持性存储区用于在断电时存储所选用户存储单元的值。 发生掉电时，CPU 留出了足够的缓冲时间来保存几个有限的指定单元的值。 这些保持性值随后在上电时进行恢复。

要显示当前项目的存储器使用情况，请右键单击相应 CPU（或其中的某个块），然后从上下文菜单中选择“资源”(Resources)。 要显示当前 CPU 的存储器使用情况，请双击“在线和诊断”(Online and diagnostics)，展开“诊断”(Diagnostics)，然后选择“存储器”(Memory)。

保持性存储器

通过将某些数据标记为保持性数据可以避免在出现电源故障后导致数据丢失。 以下数据可以组态为保持性数据：

- 位存储器 (M)： 可以在 PLC 变量表或分配列表中定义位存储器的具体存储器宽度。 保持性位存储器总是从 MB0 开始向上连续贯穿指定的字节数。 通过 PLC 变量表或在分配列表中通过单击“保持性”(Retain) 工具栏图标指定该值。 输入从 MB0 开始保留的 M 字节个数。
- 功能块 (FB) 的变量： 如果 FB 是在选中“仅符号访问”(Symbolic access only) 框的情况下创建的，则该 FB 的接口编辑器将包括“保持性”(Retain) 列。 在该列中，可以为各变量单独选择“保持性”(Retain) 或“非保持”(Non-Retain)。 在程序编辑器中放置该 FB 时创建的背景 DB 也会显示该保持性列。但仅仅是显示，用户无法在组态为“仅符号访问”(Symbolic access only) 的 FB 的背景 DB 接口编辑器中更改保持性状态。

  如果 FB 是在未选择“仅符号访问”(Symbolic access only) 框的情况下创建的，则该 FB 的接口编辑器不会包括“保持性”(Retain) 列。 在程序编辑器中插入该 FB 时创建的背景 DB 会显示“保持性”(Retain) 列，并且该列可以编辑。 在这种情况下，为任何变量选择“保持性”(Retain) 选项都会导致选择**所有**变量。 同样，为任何变量取消选择该选项都会导致取消选择**所有**变量。 对于组态为**非**“仅符号访问”(Symbolic access only) 的 FB，可以在背景 DB 编辑器中更改保持性状态，但所有变量同时会被设置为相同的保持性状态。

  创建 FB 后，无法更改“仅符号访问”(Symbolic access only) 的选项。 该选项只能在创建 FB 时选择。 要确定现有 FB 是否组态了“仅符号访问”(Symbolic access only)，请在项目树中右键单击该 FB，选择“属性”(Properties)，然后选择“特性”(Attributes)。
- 全局数据块的变量： 在保持性状态分配方面，全局 DB 与 FB 类似。 根据符号寻址的设置情况，用户可以为全局数据块的单个变量或所有变量定义保持性状态。
  - 如果选中 DB 的“仅符号访问”(Symbolic access only) 属性，则可以为各个变量设置保持性状态。
  - 如果未选中 DB 的“仅符号访问”(Symbolic access only) 属性，则保持性状态设置将应用于该 DB 的所有变量；即变量或是都有保持性，或是都没有。

总共 2048 个字节的数据可以具有保持性。 要了解可用保持性字节数，请在 PLC 变量表或分配列表中单击“保持性”(Retain) 工具栏图标。 尽管这里是为 M 存储器指定保持性范围的地方，但第二个箭头会指示可用于 M 和 DB 的总剩余存储空间。

诊断缓冲区

CPU 支持的诊断缓冲区包含有与诊断事件一一对应的条目。 每个条目都包含了事件发生的日期和时间、事件类别及事件描述。 条目按时间顺序显示，最新发生的事件位于最上面。 在 CPU 保持通电时，该日志最多可提供 50 个最新发生的事件。 日志填满后，新事件将替换日志中最早的事件。 掉电时，将保存最新发生的十个事件。

诊断缓冲区中记录以下事件类型：

- 所有系统诊断事件；例如，CPU 错误和模块错误
- CPU 的每次状态切换（每次上电、每次切换到 STOP 模式、每次切换到 RUN 模式）

必须在线访问诊断缓冲区。 可在“在线和诊断/诊断/诊断缓冲区”(Online & diagnostics / Diagnostics / Diagnostics buffer) 下查找该日志。 有关故障排除和调试的更多信息，请参考“在线和诊断”章节。

日时钟

CPU 支持日时钟。 在 CPU 断电期间，超级电容器提供时钟继续运行所需的电能。 超级电容器在 CPU 通电时充电。 在 CPU 通电至少 2 小时之后，超级电容器所具有的电量通常足以维持时钟运行 10 天。

日时钟被设置为系统时间，该时间是协调世界时 (UTC, Coordinated Universal Time)。 STEP 7 Basic 将日时钟设置为系统时间。 有相关指令用于读取系统时间 (RD_SYS_T) 或本地时间 (RD_LOC_T)。 通过使用用户在 CPU 时钟设备配置中设置的时区和夏令时偏移量计算本地时间。

可在“日时钟”(Time of day) 属性下组态 CPU 的日时钟。 还可以启用夏令时，并指定夏令时的开始时间和结束时间。 要设置日时钟，必须在线并且处于 CPU 的“在线和诊断”(Online & diagnostics) 视图中。 然后使用“设置日时钟”(Set time of day) 功能。

系统和时钟存储器

使用 CPU 属性可启用“系统存储器”和“时钟存储器”的相应字节。 程序逻辑可以引用这些功能的各个位。

- 可以将 M 存储器的一个字节分配给系统存储器。 系统存储器的字节提供了以下可供用户程序引用的四位：
  - “始终为 0（低）”位始终设置为 0。
  - “始终为 1（高）”位始终设置为 1。
  - “诊断图形已更改”位在 CPU 记录了诊断事件后的一个扫描周期内设置为 1。 由于直到首次程序循环 OB 执行结束，CPU 才能置位“诊断图形已更改”位，因此用户程序无法检测在启动 OB 执行期间或首次程序循环 OB 执行期间是否发生过诊断更改。
  - “首次扫描”位在启动 OB 完成后的第一次扫描期间内设置为 1。 （执行了第一次扫描后，“首次扫描”位将设置为 0。）
- 可以将 M 存储器的一个字节分配给时钟存储器。 被组态为时钟存储器的字节中的每一位都可生成方波脉冲。 时钟存储器字节提供了 8 种不同的频率，其范围从 0.5 Hz（慢）到 10 Hz（快）。 这些位可作为控制位（尤其在与沿指令结合使用时），用于在用户程序中周期性触发动作。

CPU 在从 STOP 模式切换到 STARTUP 模式时初始化这些字节。 时钟存储器的位在 STARTUP 和 RUN 模式下会随 CPU 时钟同步变化。

小心

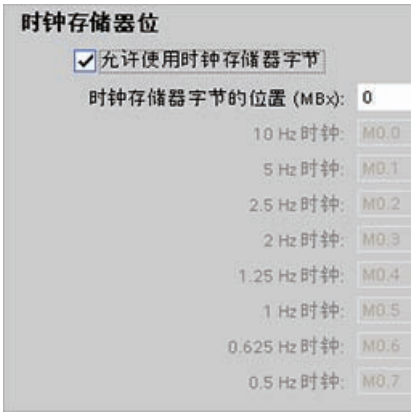
改写系统存储器或时钟存储器的各个位可能会破坏这些功能中的数据，同时还可能导致用户程序错误运行，进而造成设备损坏和人员伤害。

因为时钟存储器和系统存储器都不是预留的 M 存储器，所以指令或通信可以写入这些单元并破坏其中的数据。

避免向这些单元写入数据以确保这些功能正常运行，并且应始终为过程或机器使用紧急停止电路。

系统存储器组态了一个可在以下条件下启用（值 = 1）的字节。

- 首次扫描 (First scan)： 在运行模式下的第一个扫描周期内启用
- 诊断图形已更改 (Diagnostic graph changed)：
- 始终为 1（高）(Always 1 (high))： 始终启用
- 始终为 0（低）(Always 0 (low))： 始终禁用



时钟存储器组态了一个字节，该字节的各个位分别按固定的时间间隔循环启用和禁用。

每个时钟标志都在相应的 M 存储器位产生一个方波脉冲。 这些位可作为控制位（尤其在与沿指令结合使用时），用于在用户代码中周期性触发动作。

组态 CPU 处于 STOP 模式时的输出值特性

可以组态 CPU 处于 STOP 模式时数字量输出和模拟量输出的特性。 可以将 CPU、SB 或 SM 的任何输出设置为冻结值或使用替换值：

- 替换特定的输出值（默认）： 为 CPU、SB 或 SM 设备的每个输出（通道）分别输入替换值。

  数字输出通道的默认替换值为 OFF，而模拟输出通道的默认替换值为 0。

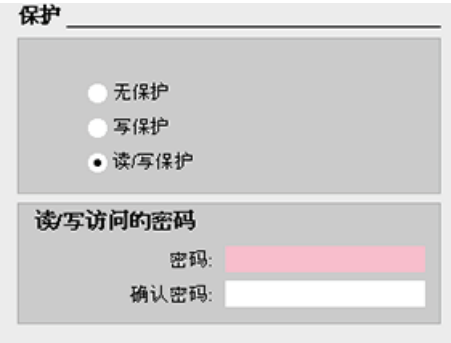
- 冻结输出以保持上一个状态： 工作模式从 RUN 切换到 STOP 时，输出将保留当前值。 上电后，输出被设置为默认的替换值。

可以在“设备配置”(Device Configuration) 中组态输出的行为。 选择相应的设备，然后使用“属性”(Properties) 选项卡组态每个设备的输出。

CPU 从 RUN 切换到 STOP 后，CPU 将保留过程映像，并根据组态写入相应的数字和模拟输出值。

3.1.4 S7-1200 CPU 的密码保护

CPU 提供了 3 个安全等级，用于限制对特定功能的访问。 为 CPU 组态安全等级和密码时，可以对那些不输入密码就能访问的功能和存储区进行限制。



要组态密码，请按以下步骤操作：

1. 在“设备配置”(Device configuration) 中，选择 CPU。
2. 在巡视窗口中，选择“属性”(Properties) 选项卡。
3. 选择“保护”(Protection) 属性以选择保护等级和输入密码。

密码区分大小写。

每个等级都允许在访问某些功能时不使用密码。 CPU 的默认状态是没有任何限制，也没有密码保护。 要限制 CPU 的访问，可以对 CPU 的属性进行组态并输入密码。

通过网络输入密码并不会使 CPU 的密码保护受到威胁。 受密码保护的 CPU 每次只允许一个用户不受限制地进行访问。 密码保护不适用于用户程序指令的执行，包括通信功能。 输入正确的密码便可访问所有功能。

PLC 到 PLC 通信（使用代码块中的通信指令）不受 CPU 中安全等级的限制。 HMI 功能同样也不受限制。

安全等级	访问限制
无保护	允许完全访问，没有密码保护。
写保护	允许 HMI 访问和各种形式的 PLC 到 PLC 通信，没有密码保护。 以下情况下需要密码：修改（写入）CPU 以及更改 CPU 模式 (RUN/STOP)。
读/写保护	允许 HMI 访问和所有形式的 PLC 到 PLC 通信，没有密码保护。 以下情况下需要密码：读取 CPU 中的数据、修改（写入）CPU 以及更改 CPU 模式 (RUN/STOP)。

3.1.5 丢失密码后恢复

如果用户丢失受密码保护的 CPU 的密码，则可使用空传送卡删除受密码保护的程序。 空传送卡将擦除 CPU 内部的装载存储器。 随后可以将新的用户程序从 STEP 7 Basic 下载到 CPU 中。

有关创建和使用空传送卡的信息，请参见传送卡 (页 69)部分。

⚠警告

如果将传送卡插入正在运行的 CPU 中，CPU 将进入 STOP 模式。控制设备在不安全情况下运行时可能会出现故障，从而导致受控设备的意外操作。这种意外运行可能会导致死亡、严重的人员伤害和/或设备损坏。

将 CPU 设置为 RUN 模式之前，必须先取出传送卡。

3.2 数据存储、存储区和寻址

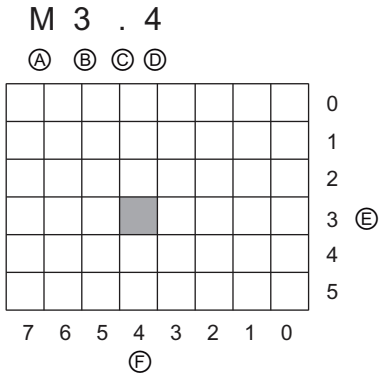
CPU 提供了以下几个选项，用于在执行用户程序期间存储数据：

- 全局储存器： CPU 提供了各种专用存储区，其中包括输入 (I)、输出 (Q) 和位存储器 (M)。 所有代码块可以无限制地访问该储存器
- 数据块 (DB)： 可在用户程序中加入 DB 以存储代码块的数据。 从相关代码块开始执行一直到结束，存储的数据始终存在。 “全局”DB 存储所有代码块均可使用的数据，而背景 DB 存储特定 FB 的数据并且由 FB 的参数进行构造。
- 临时存储器： 只要调用代码块，CPU 的操作系统就会分配要在执行块期间使用的临时或本地存储器 (L)。 代码块执行完成后，CPU 将重新分配本地存储器，以用于执行其它代码块。

每个存储单元都有唯一的地址。 用户程序利用这些地址访问存储单元中的信息。

存储区	说明	强制	保持性
I 过程映像输入 I_:P （物理输入）	在扫描周期开始时从物理输入复制	否	否
	立即读取 CPU、SB 和 SM 上的物理输入点	是	否
Q 过程映像输出 Q_:P （物理输出）	在扫描周期开始时复制到物理输出	无	否
	立即写入 CPU、SB 和 SM 上的物理输出点	是	否
M 位存储器	控制和数据存储器	否	是
L 临时存储器	存储块的临时数据，这些数据仅在该块的本地范围内有效	否	否
DB 数据块	数据存储器，同时也是 FB 的参数存储器	否	是

每个存储单元都有唯一的地址。用户程序利用这些地址访问存储单元中的信息。下图说明了如何访问一个位（也称为“字节.位”寻址）。 在此实例中，存储区和字节地址（I = 输入，而 3 = 字节 3）通过后面的句点 (.) 与位地址（位 4）分开。



A 存储区标识符

B 字节地址：字节 3

C 分隔符（“字节.位”）

D 位在字节中的位置（位 4，共 8 位）

E 存储区的字节

F 选定字节的位

通过使用“字节地址”格式，可以按字节、字或双字访问大部分存储区（I、Q、M、DB 和 L）中的数据。 要在存储器中访问数据的字节、字或双字，必须以类似于指定位地址的方式指定该地址。 该地址包括存储区标识符、数据大小标识以及字节、字或双字值的起始字节地址。 大小标识是指 B（字节）、W（字）和 D（双字），例如，IB0、MW20 或 QD8。 诸如 I0.3 和 Q1.7 等引用可访问过程映像。 要访问物理输入或输出，请在引用后面添加“:P”（例如，I0.3:P、Q1.7:P 或 "Stop:P"）。

访问 CPU 存储区中的数据

STEP 7 Basic 简化了符号编程。 通常，既可在 PLC 变量（数据块）中创建变量，也可在 OB、FC 或 FB 顶部的接口中创建变量。 这些变量包括名称、数据类型、偏移量和注释。 另外，可以在数据块中指定初始值。 在编程时，通过在指令参数中输入变量名称，可以使用这些变量。 也可以选择在指令参数中输入绝对操作数（存储器、存储区、大小和偏移量）。 以下各部分的实例介绍了如何输入绝对操作数。 程序编辑器会自动在绝对操作数前面插入 % 字符。 可以在程序编辑器中将视图切换到以下几种视图之一： 符号、符号和绝对，或绝对。

I（过程映像输入）： CPU 仅在每个扫描周期的循环 OB 执行之前对外围（物理）输入点进行采样，并将这些值写入到输入过程映像。 可以按位、字节、字或双字访问输入过程映像。 允许对过程映像输入进行读写访问，但过程映像输入通常为只读。

位	I[字节地址].[位地址]	I0.1
字节、字或双字	I[大小][起始字节地址]	IB4、IW5 或 ID12

通过在地址后面添加“:P”，可以立即读取 CPU、SB 或 SM 的数字和模拟输入。 使用 I_:P 访问与使用 I 访问的区别是，前者直接从被访问点而非输入过程映像获得数据。 这种 I_:P 访问称为“立即读”访问，因为数据是直接从源而非副本获取的，这里的副本是指在上次更新输入过程映像时建立的副本。

因为物理输入点直接从与其连接的现场设备接收值，所以不允许对这些点进行写访问。即，与可读或可写的 I 访问不同的是，I_:P 访问为只读访问。

I_:P 访问也仅限于单个 CPU、SB 或 SM 所支持的输入大小（向上取整到最接近的字节）。 例如，如果 2 DI/2 DQ SB 的输入被组态为从 I4.0 开始，则可按 I4.0:P 和 I4.1:P 形式或者按 IB4:P 形式访问输入点。不会拒绝 I4.2:P 到 I4.7:P 的访问形式，但没有任何意义，因为这些点未使用。 但不允许 IW4:P 和 ID4:P 的访问形式，因为它们超出了与该 SB 相关的字节偏移量。

使用 I_:P 访问不会影响存储在输入过程映像中的相应值。

位	I[字节地址].[位地址]:P	I0.1:P
字节、字或双字	I[大小][起始字节地址]:P	IB4:P、IW5:P 或 ID12:P

Q（过程映像输出）： CPU 将存储在输出过程映像中的值复制到物理输出点。 可以按位、字节、字或双字访问输出过程映像。 过程映像输出允许读访问和写访问。

位	Q[字节地址].[位地址]	Q1.1
字节、字或双字	Q[大小][起始字节地址]	QB5、QW10、QD40

通过在地址后面添加":P"，可以立即写入 CPU、SB 或 SM 的物理数字和模拟输出。 使用 Q_:P 访问与使用 Q 访问的区别是，前者除了将数据写入输出过程映像外还直接将数据写入被访问点（写入两个位置）。 这种 Q_:P 访问有时称为"立即写"访问，因为数据是被直接发送到目标点；而目标点不必等待输出过程映像的下一次更新。

因为物理输出点直接控制与其连接的现场设备，所以不允许对这些点进行读访问。 即，与可读或可写的 Q 访问不同的是，Q_:P 访问为只写访问。

Q_:P 访问也仅限于单个 CPU、SB 或 SM 所支持的输出大小（向上取整到最接近的字节）。 例如，如果 2 DI/2 DQ SB 的输出被组态为从 Q4.0 开始，则可按 Q4.0:P 和 Q4.1:P 形式或者按 QB4:P 形式访问输出点。不会拒绝 Q4.2:P 到 Q4.7:P 的访问形式，但没有任何意义，因为这些点未使用。 但不允许 QW4:P 和 QD4:P 的访问形式，因为它们超出了与该 SB 相关的字节偏移量。

使用 Q_:P 访问既影响物理输出，也影响存储在输出过程映像中的相应值。

位	Q[字节地址].[位地址]:P	Q1.1:P
字节、字或双字	Q[大小][起始字节地址]:P	QB5:P、QW10:P 或 QD40:P

M（位存储区）： 针对控制继电器及数据的位存储区（M 存储器）用于存储操作的中间状态或其它控制信息。 可以按位、字节、字或双字访问位存储区。 M 存储器允许读访问和写访问。

位	M[字节地址].[位地址]	M26.7
字节、字或双字	M[大小][起始字节地址]	MB20、MW30、MD50

临时（临时存储器）： CPU 根据需要分配临时存储器。 CPU 在代码块启动（对于 OB）或被调用（对于 FC 或 FB）时为其分配临时存储器。 为代码块分配临时存储器时，可能会重复使用其它 OB、FC 或 FB 先前使用的相同临时存储单元。 CPU 在分配临时存储器时不会对其进行初始化，因而临时存储器可能包含任何值。

临时存储器与 M 存储器类似，但有一个主要的区别： M 存储器在“全局”范围内有效，而临时存储器在“局部”范围内有效：

- M 存储器： 任何 OB、FC 或 FB 都可以访问 M 存储器中的数据，也就是说这些数据可以全局性地用于用户程序中的所有元素。
- 临时存储器： 只有创建或声明了临时存储单元的 OB、FC 或 FB 才可以访问临时存储器中的数据。 临时存储单元是局部有效的，并且不会被其它代码块共享，即使在代码块调用其它代码块时也是如此。 例如： 当 OB 调用 FC 时，FC 无法访问对其进行调用的 OB 的临时存储器。

CPU 为三个 OB 优先级组中的每一个都提供了临时（本地）存储器：

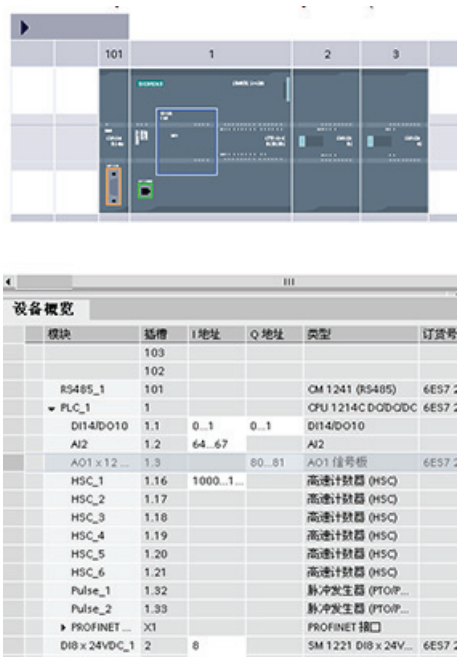
- 16 KB 用于启动和程序循环（包括相关的 FB 和 FC）
- 4 KB 用于标准中断事件（包括 FB 和 FC）
- 4 KB 用于错误中断事件（包括 FB 和 FC）

只能通过符号寻址的方式访问临时存储器。

DB（数据块）： DB 存储器用于存储各种类型的数据，其中包括操作的中间状态或 FB 的其它控制信息参数，以及许多指令（如定时器和计数器）所需的数据结构。 可以指定数据块为读/写访问还是只读访问。 可以按位、字节、字或双字访问数据块存储器。 读/写数据块允许读访问和写访问。 只读数据块只允许读访问。

位	DB[数据块编号].DBX[字节地址].[位地址]	DB1.DBX2.3
字节、字或双字	DB[数据块编号].DB [大小][起始字节地址]	DB1.DBB4、 DB10.DBW2、 DB20.DBD8

对 CPU 和 I/O 模块中的 I/O 进行寻址



向组态画面添加 CPU 和 I/O 模块时，系统会自动分配 I 地址和 Q 地址。

通过在组态画面中选择地址域并键入新编号，可以更改默认寻址设置。 数字输入和输出按完整的 8 位字节方式进行分配，无论模块是否使用所有的点。 模拟输入和输出按每组 2 点（4 个字节）的方式进行分配。 在此实例中，可以将 DI16 的地址改为 2..3 来替代 8..9。工具可以协助您更改大小错误或与其它地址相冲突的地址范围。

图中显示的实例是配有两个 SM 的 CPU 1214C。

3.3 数据类型

数据类型用于指定数据元素的大小以及如何解释数据。 每个指令参数至少支持一种数据类型，而有些参数支持多种数据类型。 将光标停在指令的参数域上方，便可看到给定参数所支持的数据类型。

形参指的是指令上标记该指令要使用的数据位置的标识符（例如，ADD 指令的 IN1 输入）。 实参指的是包含指令要使用的数据的存储单元或常量（例如，%MD400 "Number_of_Widgets"）。 用户指定的实参的数据类型必须与指令指定的形参所支持的数据类型之一匹配。

指定实参时，必须指定变量（符号）或者绝对存储器地址。 变量将符号名（变量名）与数据类型、存储区、存储器偏移量和注释关联在一起，并且可以在 PLC 变量编辑器或块（OB、FC、FB 或 DB）的接口编辑器中进行创建。 如果输入一个没有关联变量的绝对地址，使用的地址大小必须与所支持的数据类型相匹配，而默认变量将在输入时创建。

还可以为许多输入参数输入常数值。 下表列出了受支持的基本数据类型，同时还包括常量输入实例。 除 String 外，其它所有数据类型都可在 PLC 变量编辑器和块接口编辑器中使用。 String 只能在块接口编辑器中使用。 下表定义了基本数据类型。

数据类型	大小（位）	范围	常量输入实例
Bool	1	0 到 1	TRUE, FALSE, 0, 1
Byte	8	16#00 到 16#FF	16#12, 16#AB
Word	16	16#0000 到 16#FFFF	16#ABCD, 16#0001
DWord	32	16#00000000 到 16#FFFFFFFF	16#02468ACE
Char	8	16#00 到 16#FF	'A', 't', '@'
Sint	8	-128 到 127	123, -123
Int	16	-32,768 到 32,767	123, -123
Dint	32	-2,147,483,648 到 2,147,483,647	123, -123
USInt	8	0 到 255	123
UInt	16	0 到 65,535	123
UDInt	32	0 到 4,294,967,295	123
Real	32	+/-1.18 x 10^{-38} 到 +/-3.40 x 10^{38}	123.456、-3.4、-1.2E+12、3.4E-3
LReal	64	+/-2.23 x 10^{-308} 到 +/-1.79 x 10^{308}	12345.123456789 -1.2E+40
Time	32	T#-24d_20h_31m_23s_648ms 到 T#24d_20h_31m_23s_647ms 存储形式：-2,147,483,648 ms 到 +2,147,483,647 ms	T#5m_30s 5#-2d T#1d_2h_15m_30x_45ms
String	可变	0 到 254 字节字符	'ABC'

尽管以下 BCD 数字格式不能用作数据类型，但它们受转换指令支持。

格式	大小（位）	数字范围	常量输入实例
BCD16	16	-999 到 999	123, -123
BCD32	32	-9999999 到 9999999	1234567, -1234567

实数格式

如 ANSI/IEEE 754-1985 标准所述，实（或浮点）数以 32 位单精度数 (Real) 或 64 位双精度数 (LReal) 表示。 单精度浮点数的精度最高为 6 位有效数字，而双精度浮点数的精度最高为 15 位有效数字。 在输入浮点常数时，最多可以指定 6 位 (Real) 或 15 位 (LReal) 有效数字来保持精度。

计算涉及到包含非常大和非常小数字的一长串数值时，计算结果可能不准确。 如果数字相差 10 的 x 次方，其中 x > 6 (Real) 或 15 (LReal)，则会发生上述情况。 例如 (Real)：100 000 000 + 1 = 100 000 000。

字符串数据类型的格式

CPU 支持使用 STRING 数据类型存储一串单字节字符。 STRING 数据类型包含总字符数（字符串中的字符数）和当前字符数。 STRING 类型提供了多达 256 个字节，用于存储最大总字符数（1 个字节）、当前字符数（1 个字节）以及最多 254 个字符（每个字符占 1 个字节）。

可以对 IN 类型的指令参数使用带单引号的文字串（常量）。 例如，‘ABC’是由三个字符组成的字符串，可用作 S_CONV 指令中 IN 参数的输入。 还可通过在 OB、FC、FB 和 DB 的块接口编辑器中选择数据类型“字符串”来创建字符串变量。 无法在 PLC 变量编辑器中创建字符串。 在声明字符串时，可以指定最大字符串大小（单位为字节）；例如，“MyString[10]”表示为 MyString 指定的最大大小为 10 字节。 如果不包含最大大小说明符括号，则假定最大大小为 254 字节。

以下实例定义了一个最大字符数为 10 而当前字符数为 3 的 STRING。这表示该 STRING 当前包含 3 个单字节字符，但可以扩展到包含最多 10 个单字节字符。

总字符数	当前字符数	字符 1	字符 2	字符 3	...	字符 10
10	3	'C' (16#43)	'A' (16#41)	'T' (16#54)	...	-
字节 0	字节 1	字节 2	字节 3	字节 4	...	字节 11

数组

可以创建包含多个基本类型元素的数组。 数组可以在 OB、FC、FB 和 DB 的块接口编辑器中创建。 无法在 PLC 变量编辑器中创建数组。

要在块接口编辑器中创建数组，请选择数据类型“Array [lo .. hi] of type”，然后编辑“lo”、“hi”和“type”，具体如下：

- lo - 数组的起始（最低）下标
- hi - 数组的结束（最高）下标
- type - 基本数据类型之一，例如 BOOL、SINT、UDINT

下标可以为负数。 可以在块接口编辑器的“名称”(Name) 列中为数组命名。 下表列出的数组实例可能与其在块接口编辑器中显示的类似：

名称	数据类型	注释
My_Bits	Array [1 .. 10] of BOOL	该数组包含 10 个布尔值
My_Data	Array [-5 .. 5] of SINT	该数组包含 11 个 SINT 值，其中包括下标 0

可使用以下语法在程序中引用数组元素：

- Array_name[*i*], 其中，*i* 为所需下标。

以下实例可能会作为参数输入出现在程序编辑器中：

- #My_Bits[3] - 引用数组“My_Bits”的第三位
- #My_Data[-2] - 引用数组“My_Data”的第四个 SINT

符号由程序编辑器自动插入。

DTL（长格式日期和时间）数据类型

DTL 数据类型是一种 12 个字节的结构，以预定义的结构保存日期和时间信息。 可以在块的临时存储器中或者在 DB 中定义 DTL。

长度（字节）	格式	值范围	值输入的示例
12	时钟和日历 （年-月变量:小时:分钟:秒.纳秒）	最小： DTL#1970-01-01-00:00:00.0 最大： DTL#2554-12-31-23:59:59.999 999 999	DTL#2008-12-16-20:30:20.250

DTL 的每一部分均包含不同的数据类型和值范围。 指定值的数据类型必须与相应部分的数据类型相一致。

Byte	组件	数据类型	值范围
0	年	UINT	1970 到 2554
1			
2	月	USINT	1 到 12
3	日	USINT	1 到 31
4	星期几	USINT	1（星期日）到 7（星期六） 工作日不包括在值条目内。
5	小时	USINT	0 到 23
6	分	USINT	0 到 59
7	秒	USINT	0 到 59
8	纳秒	UDINT	0 到 999 999 999
9			
10			
11			

3.4 使用存储卡

注意
CPU 仅支持预格式化的 SIMATIC 存储卡 (页 372)。 如果使用 Windows 格式化程序对 SIMATIC 存储卡重新进行格式化，CPU 将无法使用该重新格式化的存储卡。 在将程序复制到格式化的存储卡之前，请删除存储卡中以前保存的所有程序。

将存储卡用作传送卡或程序卡。 复制到存储卡中的任何程序均包括所有代码块和数据块、所有工艺对象和设备配置。 程序不包含强制值。

- 使用传送卡将程序复制到 CPU 的内部装载存储器，而无需使用 STEP 7 Basic。 插入传送卡后，CPU 首先擦除内部装载存储器中的用户程序和所有强制值，然后将程序从传送卡复制到内部装载存储器。 传送过程完成后，必须取出传送卡。

  在密码丢失或忘记密码时 (页 58)，可使用空传送卡访问受密码保护的 CPU。 插入空传送卡会删除 CPU 内部装载存储器中受密码保护的程序。 随后可以将新的程序下载到 CPU 中。

- 将程序卡用作 CPU 的外部装载存储器。 在 CPU 中插入程序卡将擦除 CPU 内部装载存储器的所有内容（用户程序和所有强制值）。 CPU 然后执行外部装载存储器（程序卡）中的程序。 如果将数据下载到插有程序卡的 CPU，将仅更新外部装载存储器（程序卡）。

  因为 CPU 的内部装载存储器在插入程序卡时已被擦除，所以**必须**将程序卡保留在 CPU 上。 如果取出程序卡，CPU 将切换到 STOP 模式。 （错误 LED 闪烁，指示程序卡已取出。）

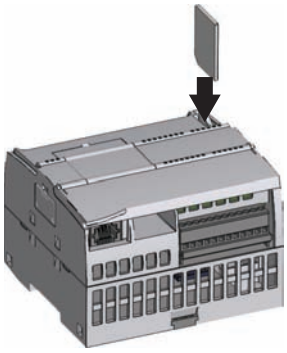
存储卡上的程序包括代码块、数据块、工艺对象和设备配置。 存储卡**不**包含任何强制值。 强制值并不属于程序的组成部分，但存储在装载存储器中，也就是存储在 CPU 的内部装载存储器或者外部装载存储器（程序卡）中。 如果在 CPU 中插有程序卡，STEP 7 Basic 将仅对程序卡上的外部装载存储器应用强制值。

3.4.1 在 CPU 中插入存储卡

⚠ 警告
如果将存储卡（无论组态为程序卡还是传送卡）插入到正在运行的 CPU 中，CPU 将立即进入 STOP 模式。控制设备在不安全情况下运行时可能会出现故障，从而导致受控设备的意外操作。这种意外运行可能会导致死亡、严重的人员伤害和/或设备损坏。 因此务必要为您的应用或过程安装急停电路。

小心
静电放电可能会损坏存储卡或 CPU 上的卡槽。 在操控存储卡时，请接触接地传导垫和/或佩戴接地腕带。 将存储卡存放在导电容器内。

要插入存储卡，需打开 CPU 顶盖，然后将存储卡插入到插槽中。推弹式连接器可以轻松地插入和取出。存储卡要求正确安装。



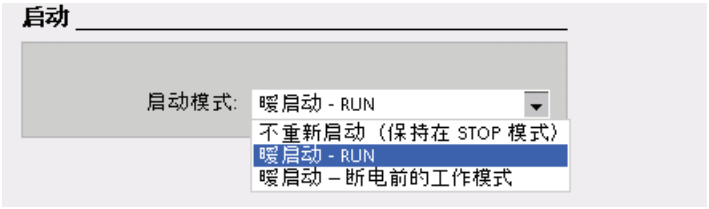
检查以确定存储卡没有写保护。 滑动保护开关，使其离开“Lock”位置。

说明

如果在 CPU 处于 STOP 模式时插入存储卡，则诊断缓冲区将显示一条消息提示存储卡评估已经启动。请忽略此消息。 直到将 CPU 切换到 RUN 模式、使用 MRES 复位 CPU 存储器或者 CPU 循环上电后，存储卡评估才会启动。

3.4.2 将项目复制到存储卡之前组态 CPU 的启动参数

将程序复制到传送卡或程序卡时，程序中包含了 CPU 的启动参数。 将程序复制到传送卡之前，请始终确保组态了 CPU 在循环上电后的工作模式。选择 CPU 是在 STOP 模式、RUN 模式还是上一个模式（通电周期之前）下启动。



3.4.3 传送卡

小心
静电放电可能会损坏存储卡或 CPU 上的卡槽。 在操控存储卡时，请先接触接地传导垫和/或佩戴接地腕带。将存储卡存放在导电容器内。

创建传送卡

请务必牢记在将程序复制到存储卡之前组态 CPU 的启动参数 (页 69)。 要创建传送卡，请按以下步骤操作：

1. 将空存储卡插入与编程设备相连的读卡器/写卡器中。

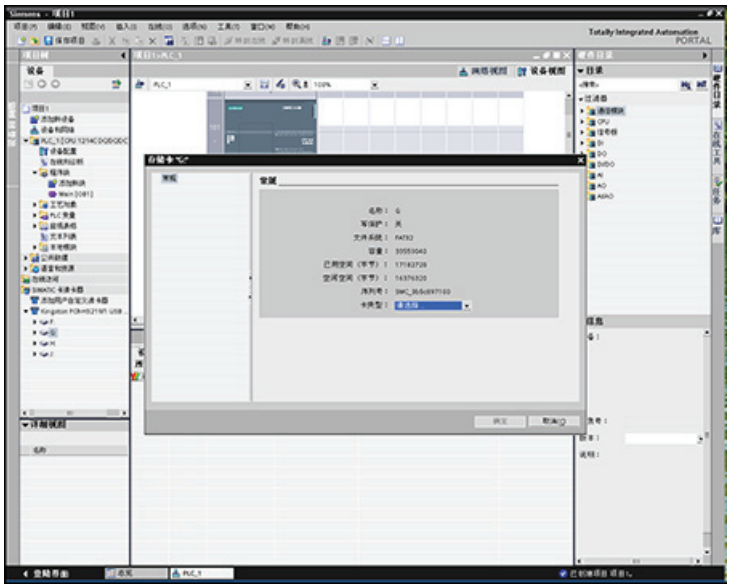
   （如果存储卡不是空卡，请使用 Windows 资源管理器之类的应用程序删除存储卡上的“SIMATIC.S7S”文件夹和“S7_JOB.S7S”文件。）

2. 在项目树中（项目视图），展开“SIMATIC 卡读卡器”(SIMATIC Card Reader) 文件夹，然后选择读卡器。

3. 右键单击读卡器中的存储卡，然后从上下文菜单中选择“属性”(Properties)，显示“存储卡”(Memory card) 对话框。

4. 在“存储卡”(Memory card) 对话框中，从下拉菜单中选择“传送”(Transfer)。

   此时，STEP 7 Basic 会创建空传送卡。 如果要创建空传送卡以便在丢失 CPU 密码 (页 58)后恢复，请从读卡器中移除传送卡。



5. 通过在项目树中选择 CPU 设备（例如 PLC_1 [CPU 1214 DC/DC/DC]），将该 CPU 设备拖动到存储卡来添加程序。 （另一种方法是复制 CPU 设备，并将其粘贴到存储卡中。） 将 CPU 设备复制到存储卡时，“装载预览”(Load preview) 对话框会打开。

6. 在“装载预览”(Load preview) 对话框中，单击“装载”(Load) 按钮，以将 CPU 设备复制到存储卡。

7. 在对话框显示一条消息指示 CPU 设备（程序）已正确装载时，单击“完成”(Finish) 按钮。

使用传送卡

要将程序传送到 CPU，请按以下步骤操作：

1. 将传送卡插入 CPU 中 (页 68)。 如果 CPU 处于 RUN 模式，它将转至 STOP 模式。（维护 LED 闪烁，指示需要对存储卡进行评估。）
2. 使用下列方法之一评估存储卡：
   - 对 CPU 循环上电。
   - 执行 STOP 到 RUN 切换。
   - 执行存储器复位 (MRES)。
3. 重启并对存储卡进行评估后，CPU 会将程序复制到内部装载存储器。 复制操作完成后，CPU 的维护 LED 闪烁，指示可以取出传送卡。
4. 从 CPU 中取出“传送”卡。
5. 使用下列方法之一评估传送到内部存储器的新程序：
   - 对 CPU 循环上电。
   - 执行 STOP 到 RUN 切换。
   - 执行存储器复位 (MRES)。

CPU 随后进入您为项目组态的启动模式（RUN 或 STOP）。

说明

将 CPU 设置为 RUN 模式之前，必须先取出传送卡。

3.4.4 程序卡

小心
静电放电可能会损坏存储卡或 CPU 上的卡槽。 在操控存储卡时，请先接触接地传导垫和/或佩戴接地腕带。将存储卡存放在导电容器内。

检查以确定存储卡没有写保护。 滑动保护开关，使其离开“Lock”位置。

在将程序元素复制到程序卡之前，请删除存储卡中以前保存的所有程序。

创建程序卡

存储卡被用作程序卡时，它就是 CPU 的外部装载存储器。 如果取出程序卡，CPU 的内部装载存储器会是空的。

说明

如果在 CPU 中插入空存储卡，然后通过对 CPU 循环上电、执行 STOP 到 RUN 切换或者执行存储器复位 (MRES) 来进行储存卡评估，则 CPU 内部装载存储器中的程序和强制值将复制到存储卡中。 （此时存储卡就是程序卡。） 复制完成后，将擦除 CPU 内部装载存储器中的程序。 CPU 随后进入组态的启动模式（RUN 或 STOP）。

请务必牢记在将项目复制到程序卡之前组态 CPU 的启动参数 (页 69)。 要使用 STEP 7 Basic 创建程序卡，请按以下步骤操作：

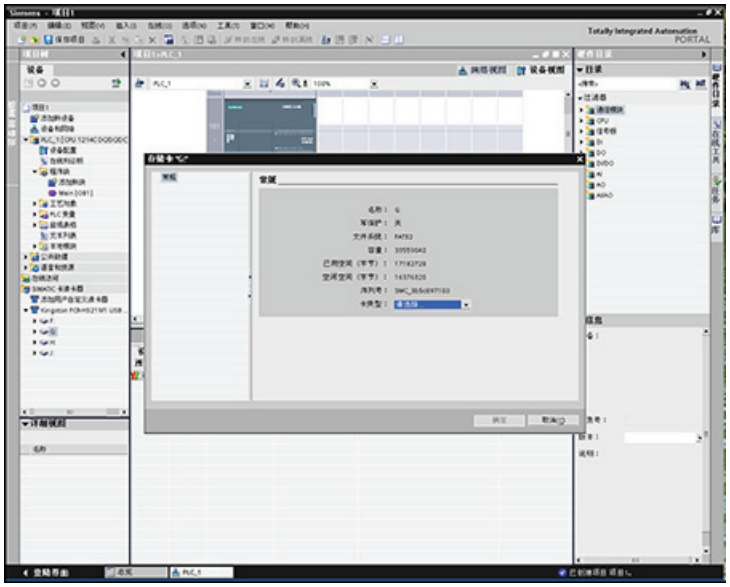
1. 将空存储卡插入与编程设备相连的读卡器/写卡器中。

   （如果存储卡不是空卡，请使用 Windows 资源管理器之类的应用程序删除存储卡上的“SIMATIC.S7S”文件夹和“S7_JOB.S7S”文件。）

2. 在项目树中（项目视图），展开“SIMATIC 卡读卡器”(SIMATIC Card Reader) 文件夹，然后选择读卡器。

3. 右键单击读卡器中的存储卡，然后从上下文菜单中选择“属性”(Properties)，显示“存储卡”(Memory card) 对话框。

4. 在“存储卡”(Memory card) 对话框中，从下拉菜单中选择“程序”(Program)。



5. 通过在项目树中选择 CPU 设备（例如 PLC_1 [CPU 1214 DC/DC/DC]），将该 CPU 设备拖动到存储卡来添加程序。（另一种方法是复制 CPU 设备，并将其粘贴到存储卡中。）将 CPU 设备复制到存储卡时，“装载预览”(Load preview) 对话框会打开。
6. 在“装载预览”(Load preview) 对话框中，单击“装载”(Load) 按钮，以将 CPU 设备复制到存储卡。
7. 在对话框显示一条消息指示 CPU 设备（程序）已正确装载时，单击“完成”(Finish) 按钮。

将程序卡用作 CPU 的装载存储器

小心
如果将空存储卡插入 CPU 中，CPU 将进入 STOP 模式。 如果对 CPU 循环上电、将 CPU 从 STOP 模式切换到 RUN 模式，或者复位 CPU 存储器 (MRES)，则 CPU 会将其内部装载存储器上的程序复制到存储卡（存储卡被组态为程序卡），并擦除内部装载存储器中的程序。 如果取出程序卡，CPU 的内部装载存储器中将没有任何程序。

要对 CPU 使用程序卡，请按以下步骤操作：

1. 将程序卡插入 CPU。 如果 CPU 处于 RUN 模式，则它将切换到 STOP 模式。 维护 LED 闪烁，指示需要对程序卡进行评估。
2. 使用下列方法之一评估程序卡：
   - 对 CPU 循环上电。
   - 执行 STOP 到 RUN 切换。
   - 执行存储器复位 (MRES)。
3. CPU 自身将重启。 重启并对程序卡进行评估后，CPU 将擦除其内部装载存储器。

CPU 随后进入您为 CPU 组态的启动模式（RUN 或 STOP）。

程序卡必须保留在 CPU 中。 取出程序卡将导致 CPU 的内部装载存储器中不会留下任何程序。

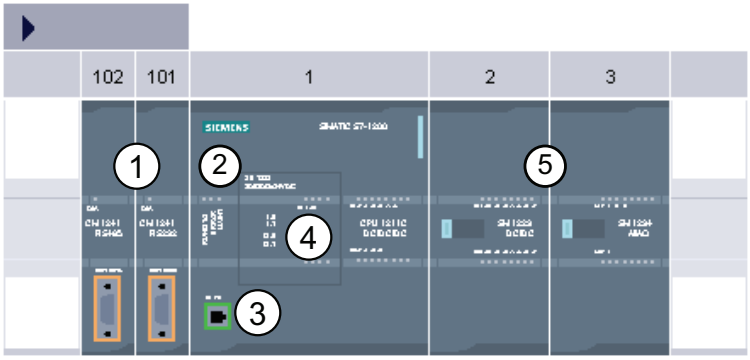
⚠警告

如果取出程序卡，CPU 将失去外部装载存储器，并生成一条错误消息。 CPU 切换到 STOP 模式并且错误 LED 闪烁。

控制设备在不安全情况下运行时可能会出现故障，从而导致受控设备的意外操作。这种意外运行可能会导致死亡、严重的人员伤害和/或设备损坏。

4 设备配置

通过向项目中添加 CPU 和其它模块，可以为 PLC 创建设备配置。



① 通信模块 (CM)： 最多 3 个，分别插在插槽 101、102 和 103 中

② CPU： 插槽 1

③ CPU 的以太网端口

④ 信号板 (SB)：最多 1 个，插在 CPU 中

⑤ 数字或模拟 I/O 的信号模块 (SM)：最多 8 个，分别插在插槽 2 到 9 中
（CPU 1214C 允许使用 8 个；CPU 1212C 允许使用 2 个；CPU 1211C 不允许使用任何信号模块）

要创建设备配置，需向项目中添加设备。

- 在门户视图中，选择“设备和网络”(Devices & Networks) 并单击“添加设备”(Add device)。

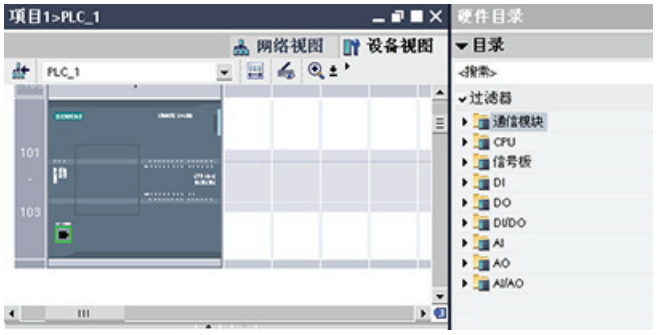
- 在项目视图中的项目名称下，双击“添加新设备”(Add new device)。

4.1 插入 CPU

通过将 CPU 插入到项目中，可创建设备配置。 通过从“添加新设备”(Add a new device) 对话框中选择 CPU，可创建机架和 CPU。

“添加新设备”对话框

硬件配置的设备视图



通过在设备视图中选择 CPU，可在巡视窗口中显示 CPU 属性。

说明

CPU 不具有预组态的 IP 地址。 设备配置期间必须为 CPU 手动分配 IP 地址。 如果 CPU 连接到网络上的路由器，则也应输入路由器的 IP 地址。

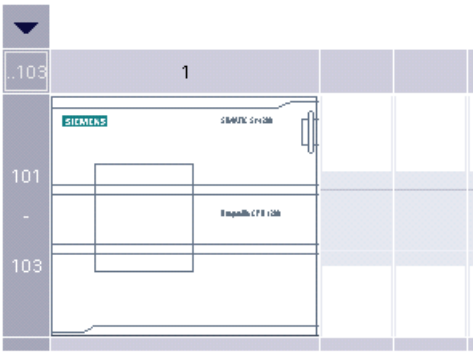
4.2 检测未指定的 CPU 的组态

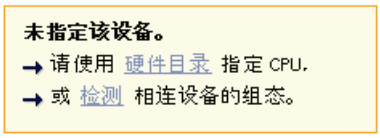
上传现有硬件配置非常简单

如果已连接到 CPU，则可以将该 CPU（包括所有模块）的组态上传到用户项目中。 只需创建新项目并选择“未指定的 CPU”而不是选择特定的 CPU 即可。 （也可通过从“新手上路”(First steps) 中选择“创建 PLC 程序”(Create a PLC program) 完全跳过设备配置。 STEP 7 Basic 即会自动创建一个未指定的 CPU。）

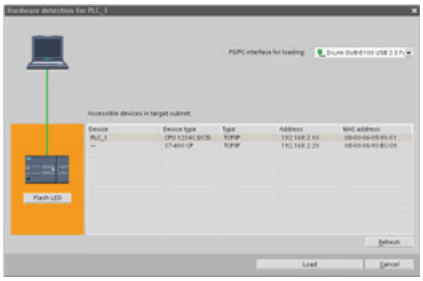
在程序编辑器中，从“在线”(Online) 菜单中选择“硬件检测”(Hardware detection) 命令。

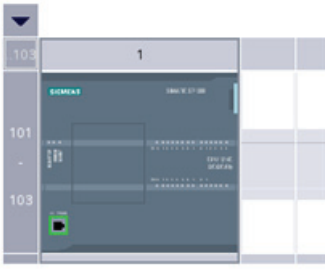
在设备配置编辑器中，选择用于检测所连设备组态的选项。





从在线对话框中选择 CPU 之后，STEP 7 Basic 会上传 CPU 以及所有模块（SM、SB 或 CM）的硬件配置。 随后可以为 CPU 和模块组态参数。





4.3 组态 CPU 的运行

要组态 CPU 的运行参数，在设备视图（整个 CPU 周围的蓝色轮廓）中选择 CPU，并使用巡视窗口的“属性”(Properties) 选项卡。

编辑属性以组态以下参数：

- PROFINET 接口： 设置 CPU 的 IP 地址和时间同步
- DI、DO 和 AI： 组态本地（板载）数字和模拟 I/O 的特性
- 高速计数器和脉冲发生器： 启用并组态高速计数器 (HSC, High-Speed Counter) 以及用于脉冲串运行 (PTO, Pulse-Train Operation) 和脉冲宽度调制 (PWM, Pulse-Width Modulation) 的脉冲发生器

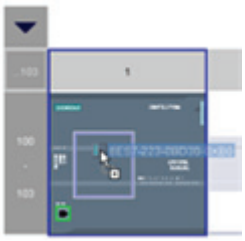
  将 CPU 或信号板的输出组态为脉冲发生器时（供 PWM 或基本运动控制指令使用），这会从 Q 存储器中移除相应的输出地址（Q0.0、Q0.1、Q4.0 和 Q4.1），并且这些地址在用户程序中不能用于其它用途。如果用户程序向用作脉冲发生器的输出写入值，则 CPU 不会将该值写入到物理输出。
- 启动： 选择进行开关转换之后 CPU 的特性，如在 STOP 模式下启动或在暖启动后转到 RUN 模式
- 日时钟： 设置时间、时区和夏令时
- 保护： 设置用于访问 CPU 的读/写保护和密码
- 系统和时钟存储器： 启用一个字节用于“系统存储器”功能（用于“首次扫描”位、“始终打开”位和“始终关闭”位），并启用一个字节用于“时钟存储器”功能（其中每个位都按预定义频率打开和关闭）。
- 循环时间： 定义最大循环时间或固定的最小循环时间
- 通信负载： 分配专门用于通信任务的 CPU 时间百分比

4.4 将模块添加到组态

使用硬件目录将模块添加到 CPU。 有三种类型的模块：

- 信号模块 (SM) 提供附加的数字或模拟 I/O 点。 这些模块连接在 CPU 右侧。
- 信号板 (SB) 仅为 CPU 提供几个附加的 I/O 点。 SB 安装在 CPU 的前端。
- 通信模块 (CM) 为 CPU 提供附加的通信端口（RS232 或 RS485）。 这些模块连接在 CPU 左侧。

要将模块插入到硬件配置中，可在硬件目录中选择模块，然后双击该模块或将其拖到高亮显示的插槽中。

模块	选择模块	插入模块	结果
SM			
SB			
CM			

4.5 组态模块的参数

要组态模块的运行参数，请在设备视图中选择模块，并使用巡视窗口的“属性”(Properties)选项卡组态模块的参数。

组态信号模块 (SM) 或信号板 (SB)

- 数字量 I/O： 可组态各个输入用于上升沿检测或下降沿检测（将每个检测分别与一个事件和硬件中断进行关联），并用于在输入过程映像的下一次更新期间进行“脉冲捕捉”（瞬时脉冲之后停留）输出可使用冻结值或替换值。
- 模拟量 I/O： 为各个输入组态参数，如测量类型（电压或电流）、范围和平滑化，也可启用下溢或上溢诊断。 输出提供诸如输出类型（电压或电流）之类的参数，也可用于诊断，例如，短路（针对电压输出）或上/下限诊断
- I/O 诊断地址： 组态用于设置模块的输入和输出的起始地址

组态通信模块 (CM)

- 端口组态： 组态通信参数，如波特率、奇偶校验、数据位、停止位、流控制、XON 和 XOFF 字符以及等待时间
- 发送消息组态： 启用和组态发送相关的选项
- 接收消息组态： 启用和组态消息起始参数和消息结束参数

这些组态参数可以由程序进行更改。

4.6 创建网络连接

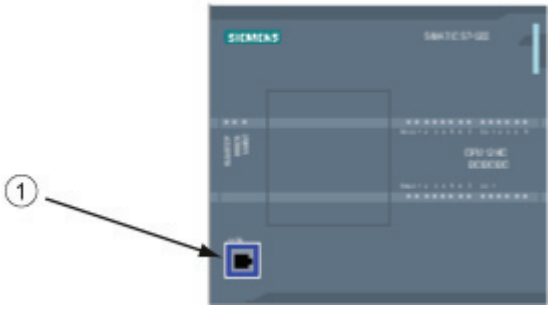
使用设备配置的“网络视图”(Network view) 在项目中的各个设备之间创建网络连接。 创建网络连接之后，使用巡视窗口的“属性”(Properties) 选项卡组态网络的参数。

操作	结果
选择“网络视图”(Network view) 以显示要连接的设备。	
选择一个设备上的端口，然后将连接拖到第二个设备上的端口处。	
释放鼠标按钮以创建网络连接。	

4.7 在项目中组态 IP 地址

组态 PROFINET 接口

使用 CPU 配置机架 (页 78)之后，可组态 PROFINET 接口的参数。 为此，单击 CPU 上的绿色 PROFINET 框以选择 PROFINET 端口。 巡视窗口中的“属性”(Properties) 选项卡会显示 PROFINET 端口。



① PROFINET 端口

组态 IP 地址

以太网 (MAC) 地址： 在 PROFINET 网络中，制造商会为每个设备都分配一个“介质访问控制”地址（MAC 地址）以进行标识。 MAC 地址由六组数字组成，每组两个十六进制数，这些数字用连字符 (-) 或冒号 (:) 分隔并按传输顺序排列（例如 01-23-45-67-89-AB 或 01:23:45:67:89:AB）。

IP 地址： 每个设备也都必须具有一个 Internet 协议 (IP) 地址。 该地址使设备可以在更加复杂的路由网络中传送数据。

每个 IP 地址分为四段，每段占 8 位，并以点分十进制格式表示（例如，211.154.184.16）。 IP 地址的第一部分用于表示网络 ID（您正位于什么网络中？），地址的第二部分表示主机 ID（对于网络中的每个设备都是唯一的）。 IP 地址 192.168.x.y 是一个标准名称，视为未在 Internet 上路由的专用网的一部分。

子网掩码： 子网是已连接的网络设备的逻辑分组。 在局域网 (LAN, Local Area Network) 中，子网中的节点往往彼此之间的物理位置相对接近。 掩码（称为子网掩码或网络掩码）定义 IP 子网的边界。

子网掩码 255.255.255.0 通常适用于小型本地网络。 这就意味着此网络中的所有 IP 地址的前 3 个八位位组应该是相同的，该网络中的各个设备由最后一个八位位组（8 位域）来标识。 举例来说，在小型本地网络中，为设备分配子网掩码 255.255.255.0 和 IP 地址 192.168.2.0 到 192.168.2.255。

不同子网间的唯一连接通过路由器实现。 如果使用子网，则必须部署 IP 路由器。

IP 路由器： 路由器是 LAN 之间的链接。 通过使用路由器，LAN 中的计算机可向其它任何网络发送消息，这些网络可能还隐含着其它 LAN。 如果数据的目的地不在 LAN 内，路由器会将数据转发给可将数据传送到其目的地的另一个网络或网络组。

路由器依靠 IP 地址来传送和接收数据包。

IP 地址属性： 在“属性”(Properties) 窗口中，选择“以太网地址”(Ethernet address) 组态条目。 TIA 门户将显示以太网地址组态对话框，该对话框可将软件项目与接收该项目的 CPU 的 IP 地址相关联。

说明

CPU 不具有预组态的 IP 地址。 必须手动为 CPU 分配 IP 地址。 如果 CPU 连接到网络上的路由器，则也必须输入路由器的 IP 地址。 下载项目时会组态所有 IP 地址。
更多相关信息，请参见“为编程设备和网络设备分配 IP 地址 (页 246)”。

下表定义了 IP 地址的参数：

参数		说明
子网	连接到设备的子网的名称。 单击“添加新子网”(Add new subnet) 按钮以创建新的子网。 默认设置为“未连接”(Not connected)。 有两种连接类型可用： • 默认情况下“未连接”(Not connected) 提供本地连接。 • 网络具有两个或多个设备时，需要子网。	
IP 协议	IP 地址	为 CPU 分配的 IP 地址
	子网掩码	分配的子网掩码
	使用 IP 路由器	单击该复选框以指示 IP 路由器的使用
	路由器地址	为路由器分配的 IP 地址（如果适用）

5 编程概念

5.1 设计 PLC 系统的指南

设计 PLC 系统时，可从若干方法和标准中进行选择。 下列常规指南可应用到许多设计项目中。 当然，必须遵守您自己公司程序的指令、自身培训以及当地已被接受的实践。

建议步骤	任务
对过程或机器进行分区	将过程或机器划分为彼此独立的部分。 这些分区会确定控制器之间的边界，并影响功能描述规范和资源的分配。
创建功能规范	写下过程或机器的每一部分（如 I/O 点）的操作说明、操作的功能描述、在允许进行每个执行器（如螺线管、电机或驱动器）的操作之前必须实现的状态、操作员界面的描述以及过程或机器其它部分的任何接口。
设计安全电路	出于安全考虑，标识任何可能需要硬接线逻辑的设备。 请记住，控制设备在不安全方式下可能会出现故障，可能会造成意外启动或机械运转变化。 其中意外或错误的机械运转可能会导致人员的身体伤害或重大的财产损失，请考虑实施机电替代装置（其独立于 PLC 运行）以防止不安全的运行。 安全电路的设计中应包含以下任务： • 标识任何可能造成危险的不正确或意外的执行器操作。 • 标识可确保操作不危险的条件，并确定如何独立于 PLC 检测这些条件。 • 标识上电和断电时 PLC 如何影响过程，并标识检测错误的方式和时间。 此信息仅用于设计正常和预期的异常操作。 出于安全考虑，不应依赖此“最佳情况”方案。 • 设计可独立于 PLC 来阻止危险运行的手动或机电安全替代装置。 • 从独立于 PLC 的电路提供相应状态信息，以便程序和任何操作员界面具有必要的信息。 • 标识针对过程安全运行的任何其它安全相关要求。
指定操作员站	根据功能规范的要求，创建以下操作员站的绘图： • 显示与过程或机器相关的每个操作员站的位置的总览图。 • 操作员站中设备的机械布局图，如显示屏、开关和灯。 • 包含 PLC 和信号模块中相关 I/O 的电气图。

建议步骤	任务
创建组态图	根据功能规范的要求，创建控制设备的组态图： • 显示与过程或机器相关的每个 PLC 位置的总览图。 • 每个 PLC 和任何 I/O 模块的机械布局图，其中包括任何控制柜及其它设备。 • 每个 PLC 和任何 I/O 模块的电气图，其中包括设备模型号、通信地址和 I/O 地址。
创建符号名称的列表	创建绝对地址的符号名称列表。 不仅包括物理 I/O 信号，也包括要在程序中使用的其它元素（如变量名）。

5.2 构建用户程序

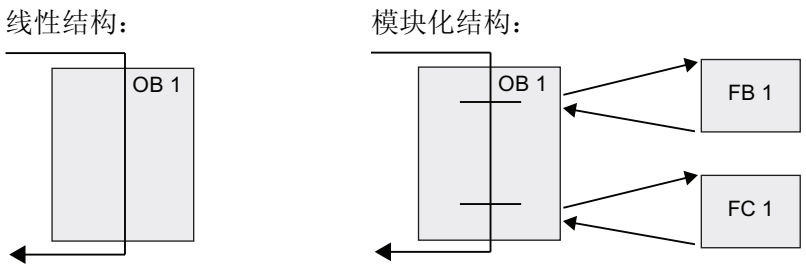
创建用于自动化任务的用户程序时，需要将程序的指令插入代码块中：

- 组织块 (OB) 对应于 CPU 中的特定事件，并可中断用户程序的执行。 用于循环执行用户程序的默认组织块 (OB 1) 为用户程序提供基本结构，是唯一一个用户必需的代码块。 如果程序中包括其它 OB，这些 OB 会中断 OB 1 的执行。其它 OB 可执行特定功能，如用于启动任务、用于处理中断和错误或者用于按特定的时间间隔执行特定的程序代码。
- 功能块 (FB) 是从另一个代码块（OB、FB 或 FC）进行调用时执行的子例程。 调用块将参数传递到 FB，并标识可存储特定调用数据或该 FB 实例的特定数据块 (DB)。 更改背景 DB 可使通用 FB 控制一组设备的运行。 例如，借助包含每个泵或阀门的特定运行参数的不同背景 DB，一个 FB 可控制多个泵或阀。
- 功能 (FC) 是从另一个代码块（OB、FB 或 FC）进行调用时执行的子例程。 FC 不具有相关的背景 DB。 调用块将参数传递给 FC。 FC 中的输出值必须写入存储器地址或全局 DB 中。

为用户程序选择结构类型

根据实际应用要求，可选择线性结构或模块化结构用于创建用户程序：

- 线性程序按顺序逐条执行用于自动化任务的所有指令。 通常，线性程序将所有程序指令都放入用于循环执行程序的 OB (OB 1) 中。
- 模块化程序调用可执行特定任务的特定代码块。 要创建模块化结构，需要将复杂的自动化任务划分为与过程的工艺功能相对应的更小的次级任务。 每个代码块都为每个次级任务提供程序段。 通过从另一个块中调用其中一个代码块来构建程序。



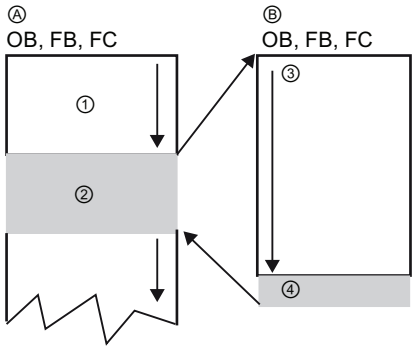
通过创建可在用户程序中重复使用的通用代码块，可简化用户程序的设计和实现。 使用通用代码块具有许多优点：

- 可为标准任务创建能够重复使用的代码块，如用于控制泵或电机。 也可以将这些通用代码块存储在可由不同的应用或解决方案使用的库中。
- 将用户程序构建到与功能任务相关的模块化组件中，可使程序的设计更易于理解和管理。 模块化组件不仅有助于标准化程序设计，也有助于使更新或修改程序代码更加快速和容易。
- 创建模块化组件可简化程序的调试。 通过将整个程序构建为一组模块化程序段，可在开发每个代码块时测试其功能。
- 创建与特定工艺功能相关的模块化组件，有助于简化对已完成应用程序的调试，并减少调试过程中所用的时间。

5.3 使用块来构建程序

通过设计 FB 和 FC 执行通用任务，可创建模块化代码块。 然后可通过由其它代码块调用这些可重复使用的模块来构建程序。 调用块将设备特定的参数传递给被调用块。

A 调用块
B 被调用（或中断）块
① 程序执行
② 可调用其它块的操作
③ 程序执行
④ 块结束（返回到调用块）



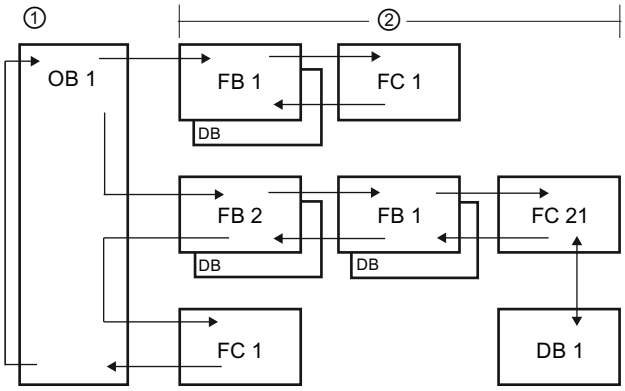
当一个代码块调用另一个代码块时，CPU 会执行被调用块中的程序代码。 执行完被调用块后，CPU 会继续执行调用块。

继续执行该块调用之后的指令。

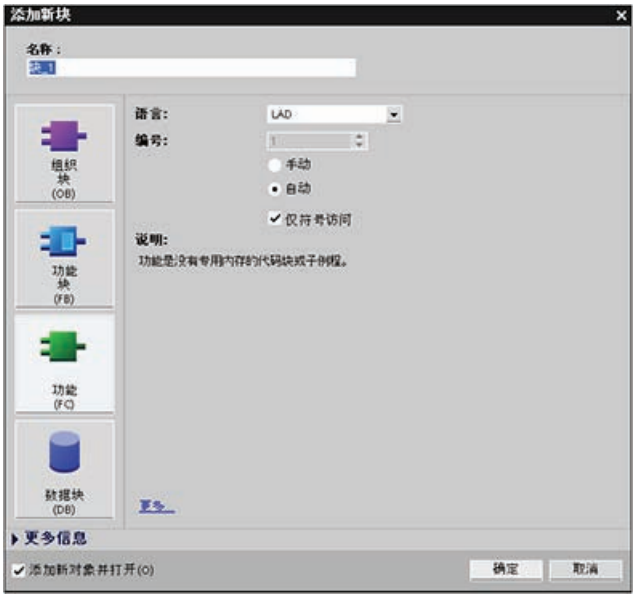
可嵌套块调用以实现更加模块化的结构。

① 循环开始

② 嵌套深度



创建可重复使用的代码块



使用项目浏览器中“程序块”(Program blocks) 下的“添加新块”(Add new block) 对话框创建 OB、FB、FC 和全局 DB。

创建代码块时，需要为块选择编程语言。 无需为 DB 选择语言，因为它仅用于存储数据。

5.3.1 组织块 (OB)

组织块为程序提供结构。 它们充当操作系统和用户程序之间的接口。 OB 是由事件驱动的。 事件（如诊断中断或时间间隔）会使 CPU 执行 OB。某些 OB 预定义了起始事件和行为。

程序循环 OB 包含用户主程序。用户程序中可包含多个程序循环 OB。RUN 模式期间，程序循环 OB 以最低优先级等级执行，可被其它各种类型的程序处理中断。 启动 OB 不会中断程序循环 OB，因为 CPU 在进入 RUN 模式之前将先执行启动 OB。

完成程序循环 OB 的处理后，CPU 会立即重新执行程序循环 OB。 该循环处理是用于可编程逻辑控制器的“正常”处理类型。 对于许多应用来说，整个用户程序位于一个程序循环 OB 中。

可创建其它 OB 以执行特定的功能，如启动任务、用于处理中断和错误或用于以特定的时间间隔执行特定程序代码。 这些 OB 会中断程序循环 OB 的执行。

使用“添加新块”(Add new block) 对话框在用户程序中创建新的 OB。

根据其相应的优先级等级，一个 OB 可中断另一个 OB。中断处理总是由事件驱动的。 发生此类事件时，CPU 会中断用户程序的执行并调用已组态用于处理该事件的 OB。 完成中断 OB 的执行后，CPU 会在中断点继续执行用户程序。

CPU 根据分配给各个 OB 的优先级来确定中断事件的处理顺序。每个事件都具有一个特定的处理优先级。 多个中断事件可合并为优先级等级。 更多相关信息，请参见 PLC 概念一章，执行用户程序小节 (页 39)。

在某等级的 OB 内创建附加 OB

可为用户程序，甚至为程序循环和启动 OB 等级创建多个 OB。 使用“添加新块”(Add new block) 对话框创建 OB。输入 OB 的名称以及一个大于 200 的 OB 编号。

如果为用户程序创建了多个程序循环 OB，则 CPU 会按数字顺序从主程序循环 OB（默认为 OB 1）开始执行每个程序循环 OB。例如： 当第一个程序循环 OB (OB 1) 完成后，CPU 将执行第二个程序循环 OB（例如 OB 200）。

组态 OB 的运行

可修改 OB 的运行参数。例如，可为延时 OB 或循环 OB 组态时间参数。

5.3.2 功能 (FC)

功能 (FC) 是通常用于对一组输入值执行特定运算的代码块。 FC 将此运算结果存储在存储器位置。

使用 FC 可执行以下任务：

- 执行标准和可重复使用的运算，如数学计算。
- 执行工艺功能，如通过使用位逻辑运算进行单独控制。

FC 也可以在程序中的不同位置多次调用。 此重复使用简化了对经常重复发生的任务的编程。

FC 不具有相关的背景数据块 (DB)。 对于用于计算该运算的临时数据，FC 采用了局部数据堆栈。 不保存临时数据。 要长期存储数据，可将输出值赋给全局存储器位置，如 M 存储器或全局 DB。

5.3.3 功能块 (FB)

功能块 (FB) 是使用背景数据块保存其参数和静态数据的代码块。 FB 具有位于数据块 (DB) 或“背景”DB 中的变量存储器。 背景 DB 提供与 FB 的实例（或调用）关联的一块存储区并在 FB 完成后存储数据。 可将不同的背景 DB 与 FB 的不同调用进行关联。 通过背景 DB 可使用一个通用 FB 控制多个设备。 通过使一个代码块对 FB 和背景 DB 进行调用，来构建程序。 然后，CPU 执行该 FB 中的程序代码，并将块参数和静态局部数据存储在背景 DB 中。 FB 执行完成后，CPU 会返回到调用该 FB 的代码块中。 背景 DB 保留该 FB 实例的值。 随后在同一扫描周期或其它扫描周期中调用该功能块时可使用这些值。

可重复使用的代码块和关联的存储区

用户通常使用 FB 控制在一个扫描周期内未完成其运行的任务或设备的运行。 要存储运行参数以便从一个扫描快速访问到下一个扫描，用户程序中的每一个 FB 都具有一个或多个背景 DB。 调用 FB 时，也需要指定包含块参数以及用于该调用或 FB “实例”的静态局部数据的背景 DB。 FB 完成执行后，背景 DB 将保留这些值。

通过设计用于通用控制任务的 FB，可对多个设备重复使用 FB，方法是：为 FB 的不同调用选择不同的背景 DB。

FB 将输入 (IN)、输出 (OUT) 和输入/输出 (IN_OUT) 参数存储在背景 DB 中。

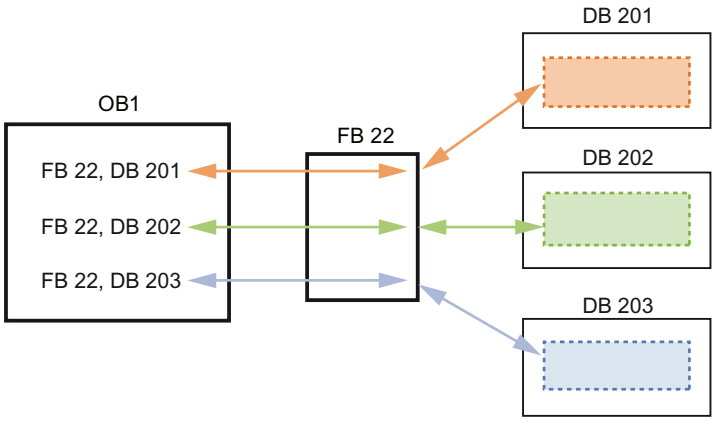
赋初值

如果没有给功能块 (FB) 的输入、输出或输入/输出参数赋值，将使用背景数据块 (DB) 中存储的值。 某些情况下，必须分配参数。

可以给 FB 接口中的参数赋初值。 这些值将传送到相关的背景 DB 中。 如果未分配参数，将使用当前存储在背景 DB 中的值。

使用带多个 DB 的单个 FB

下图显示了三次调用同一个 FB 的 OB，方法是针对每次调用使用一个不同的数据块。 该结构使一个通用 FB 可以控制多个相似的设备（如电机），方法是在每次调用时为各设备分配不同的背景数据块。 每个背景 DB 存储单个设备的数据（如速度、加速时间和总运行时间）。 在此实例中，FB 22 控制三个独立的设备，其中 DB 201 用于存储第一个设备的运行数据，DB 202 用于存储第二个设备的运行数据，DB 203 用于存储第三个设备的运行数据。



5.3.4 数据块 (DB)

在用户程序中创建数据块 (DB) 以存储代码块的数据。 用户程序中的所有程序块都可访问全局 DB 中的数据，而背景 DB 仅存储特定功能块 (FB) 的数据。 可将 DB 定义为当前只读。

相关代码块执行完成后，DB 中存储的数据不会被删除。 有两种类型的 DB：

- 全局 DB 存储程序中代码块的数据。 任何 OB、FB 或 FC 都可访问全局 DB 中的数据。
- 背景 DB 存储特定 FB 的数据。 背景 DB 中数据的结构反映了 FB 的参数（Input、Output 和 InOut）和静态数据。 （FB 的临时存储器不存储在背景 DB 中。）

说明

尽管背景 DB 反映特定 FB 的数据，然而任何代码块都可访问背景 DB 中的数据。

5.4 了解数据一致性

CPU 为所有基本数据类型（例如 Word 或 DWord）和所有系统定义的结构（例如 IEC_TIMERS 或 DTL）保持数据一致性。 值的读/写操作无法中断。 （例如，在读写四字节的 DWord 之前，CPU 会防止对该 DWord 值进行访问。） 为确保程序循环 OB 和中断 OB 无法同时写入同一个存储单元，在程序循环 OB 中的读/写操作完成之前，CPU 不会执行中断 OB。

如果用户程序共享存储器中在程序循环 OB 和中断 OB 之间生成的多个值，用户程序还必须确保在修改或读取这些值时保持一致性。 可以在程序循环 OB 中使用 DIS_AIRT 和 EN_AIRT 指令，以防止对共享值进行访问。

- 在代码块中插入 DIS_AIRT 指令，以确保在读/写操作期间无法执行中断 OB。
- 插入读/写能够被中断 OB 更改的值的指令。
- 在顺序结尾处插入 EN_AIRT 指令，以取消 DIS_AIRT，并允许执行中断 OB。

HMI 设备或另一个 CPU 发出的通信请求也能够中断程序循环 OB 的执行。通信请求也会导致与数据一致性相关的问题。 CPU 确保基本数据类型始终由用户程序指令执行一致地读取和写入。 由于通信会周期性地中断用户程序，因而不能保证 HMI 能够同时更新 CPU 中的多个值。 例如，给定 HMI 画面上显示的值可能来自 CPU 的不同扫描周期。

PtP（Point-to-Point，点对点）指令和 PROFINET 指令（例如，TSEND_C 和 TRCV_C）可用于传送可被中断的数据缓冲区。 通过避免对程序循环 OB 和中断 OB 中的缓冲区进行任何读/写操作，可以确保数据缓冲区的数据一致性。如果需要在中断 OB

中修改这些指令的缓冲区值，请使用 DIS_AIRT 指令延迟所有中断（中断 OB 或源自 HMI 或另一个 CPU 的通信中断），直到执行了 EN_AIRT 指令。

说明

使用 DIS_AIRT 指令延迟中断 OB 的处理，直到执行了 EN_AIRT 指令，以此影响用户程序的中断等待时间（从事件发生到执行中断 OB 的时间）。

5.5 选择编程语言

可以在 LAD（梯形图）或 FBD（功能块图）编程语言之间做出选择。

LAD 编程语言

LAD 是一种图形编程语言。 它使用基于电路图的表示法。

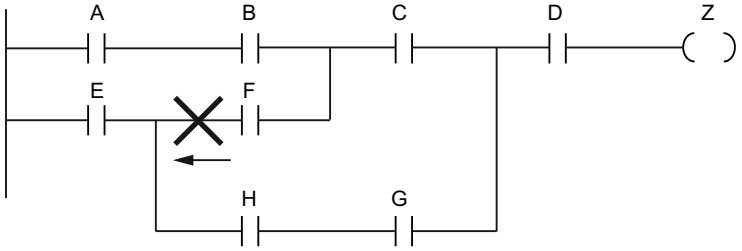
电路图的元件（如常闭触点、常开触点和线圈）相互连接构成程序段。

要创建复杂运算逻辑，可插入分支以创建并行电路的逻辑。 并行分支向下打开或直接连接到电源线。 用户可向上终止分支。

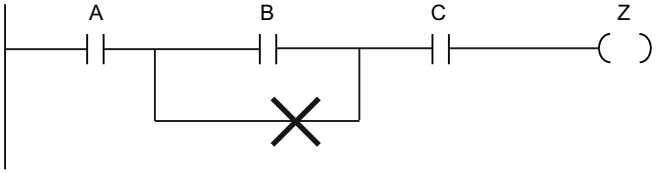
LAD 向多种功能（如数学、定时器、计数器和移动）提供“功能框”指令。

创建 LAD 程序段时请注意以下规则：

- 每个 LAD 程序段都必须使用线圈或功能框指令来终止。 不要使用比较指令或沿检测（上升沿或下降沿）指令终止程序段。
- 不能创建可能导致反向能流的分支。



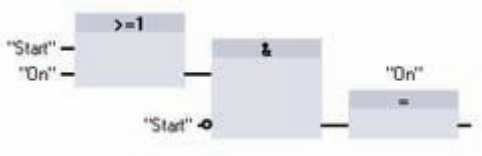
- 不能创建可能导致短路的分支。



功能块图 (FBD) 编程语言

与 LAD 一样，FBD 也是一种图形编程语言。 逻辑表示法以布尔代数中使用的图形逻辑符号为基础。

算术功能和其它复杂功能可直接结合逻辑框表示。 要创建复杂运算的逻辑，在功能框之间插入并行分支。



理解“功能框”指令的 EN 和 ENO

LAD 和 FBD 都可以将“能流”（EN 和 ENO）用于某些“功能框”指令。 特定指令（如数学和移动指令）显示 EN 和 ENO 的参数。 这些参数与能流有关并确定在该扫描期间是否执行指令。

- EN（使能输入）是 LAD 和 FBD 中功能框的布尔输入。 要执行功能框指令，能流 (EN = 1) 必须出现在此输入端。 如果 LAD 功能框的 EN 输入直接连接到左侧电源线，则将始终执行该功能框。
- ENO（使能输出）是 LAD 和 FBD 中功能框的布尔输出。 如果该功能框在 EN 输入端有能流且正确执行了其功能，则 ENO 输出会将能流 (ENO = 1) 传递到下一个元素。 如果执行功能框指令时检测到错误，则在产生该错误的功能框指令处终止该能流 (ENO = 0)。

程序编辑器	输入/输出	操作数	数据类型
LAD	EN、ENO	能流	BOOL
FBD	EN	I、I:P、Q、M、DB、Temp、能流	BOOL
	ENO	能流	BOOL

5.6 复制保护

通过复制或“专有技术”保护可防止程序中的一个或多个代码块（OB、FB 或 FC）受到未经授权的访问。 用户创建密码以限制对代码块的访问。

将块组态为“专有技术”保护时，只有在输入密码后才能访问块内的代码。

要对块实施复制保护，可从“编辑”(Edit) 菜单中选择“专有技术保护”(Know how protection) 命令。 然后输入允许访问该块的密码。

密码保护会防止对代码块进行未授权的读取或修改。 如果没有密码，只能读取有关代码块的以下信息：

- 块标题、块注释和块属性
- 传送参数（IN、OUT、IN_OUT、Return）
- 程序的调用结构
- 交叉引用中的全局变量（不带使用时的信息），但局部变量已隐藏

5.7 下载程序的元素

可将项目的元素从编程设备下载到 CPU。 下载项目时，CPU 会将用户程序（OB、FC、FB 和 DB）存储在永久存储器中。

可从以下任何位置将项目从编程设备下载到 CPU：

- “项目树”(Project tree)： 右键单击程序元素，然后单击上下文相关的“下载”(Download) 选择项。
- “在线”(Online) 菜单： 单击“下载到设备”(Download to device) 选择项。
- 工具栏： 单击“下载到设备”(Download to device) 图标。

5.8 上传程序的元素

可以将所有程序块和变量表从在线 CPU 上传到离线项目，但无法上传设备配置或监视表格。 无法上传到空项目中；必须有一个离线 CPU 可用于上传。 无法上传单个块；只能上传整个程序。 如果执行上传，则在上传前出现确认提示后将“清空”离线 CPU（删除所有块和变量表）。 用户无法在在线区域编辑块；必须先将其上传到离线区域进行修改，然后重新下载到 PLC。

有两种执行上传的方式： 在项目树中拖放，或在比较编辑器中同步。

在项目树中拖放

1. 创建一个新项目。
2. 添加与要上传的 CPU 相匹配的 CPU 设备。
3. 展开该 CPU 节点一次，以便“程序块”(Program blocks) 文件夹可见。
4. 在“项目树”中，展开“在线访问”(Online access) 节点，然后展开对应所需网络的节点并双击“更新可访问的设备”(Update accessible devices)。
5. 列出了可用的 CPU 后，展开所需 CPU 的节点。
6. 在在线访问区域中，左键单击并按住“程序块”(Program blocks) 文件夹，将其向上拖动到离线区域的“程序块”(Program blocks) 文件夹，然后松开鼠标左键。 在经过相应正确区域上方时，鼠标指针应会变为“+”。

7. 您应能看到“上传预览”(Upload preview) 对话框打开。 单击“继续”(Continue) 框，然后单击“从设备上传”(Upload from device)。
8. 等到上传完成。 此时，所有程序块、工艺块和变量应显示在离线区域中。
9. 由于无法上传设备配置，因而请使用设备配置手动设置 CPU 属性（包括所需的 IP 地址），并将其它设备添加到离线项目中。

也可以从在线区域拖动到现有程序的“程序块”(Program blocks) 区域。 也就是说，程序块离线区域并不一定是空的。 此时，现有程序将被删除，取而代之的是在线程序。

在比较编辑器中同步

1. 打开包含相应项目的项目。
2. 在“项目树”中，选择要比较的离线 CPU。
3. 右键单击离线 CPU，或从“工具”(Tools) 菜单中选择“比较离线/在线”(Compare offline/online) 命令，打开比较编辑器。
4. 比较编辑器将在“程序块”(Program blocks) 文件夹下列出不同之处。 单击动作列中的符号。 要上传项目，请选择“从设备上传”(Upload from device)。
5. 单击“同步在线和离线对象”(Synchronize online and offline) 按钮，将项目从在线 CPU 复制到离线 CPU。

5.9 调试和测试程序

使用“监视表格”监视和修改正在由在线 CPU 执行的用户程序的值。 可在项目中创建并保存不同的监视表格以支持各种测试环境。 这使得用户可以在调试期间或出于维修和维护目的重新进行测试。

通过监视表格，可监视 CPU 并与 CPU 交互，如同 CPU 执行用户程序一样。 不仅可以显示或更改代码块和数据块的变量值，也可以显示或更改 CPU 存储区（其中包括输入和输出（I 和 Q）、外围设备输入和输出（I:P 和 Q:P）、位存储器 (M) 和数据块 (DB)）的值。

通过监视表格，可在 STOP 模式下启用 CPU 的物理输出 (Q:P)。 例如，测试 CPU 的接线时可为输出端赋特定值。

监视表格也可用于“强制”变量或将变量设置为特定值。 有关强制的更多信息，请参见“在线和诊断”一章的 CPU 中的强制值 (页 316)一节。

6 编写指令

6.1 基本指令

6.1.1 位逻辑

LAD 触点

常开

常闭

可将触点相互连接并创建用户自己的组合逻辑。 如果用户指定的输入位使用存储器标识符 I（输入）或 Q（输出），则从过程映像寄存器中读取位值。 控制过程中的物理触点信号会连接到 PLC 上的 I 端子。 CPU 扫描已连接的输入信号并持续更新过程映像输入寄存器中的相应状态值。

通过在 I 偏移量后加入“:P”，可指定立即读取物理输入（例如：“%I3.4:P”）。 对于立即读取，直接从物理输入读取位数据值，而非从过程映像中读取。 立即读取不会更新过程映像。

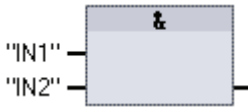
参数	数据类型	说明
IN	Bool	分配位

- 在赋的位值为 1 时，常开触点将闭合 (ON)。
- 在赋的位值为 0 时，常闭触点将闭合 (ON)。
- 以串联方式连接的触点创建 AND 逻辑程序段。
- 以并联方式连接的触点创建 OR 逻辑程序段。

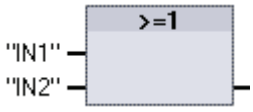
FBD、AND、OR 和 XOR 功能框

在 FBD 编程中，LAD 触点程序段变为与 (&)、或 (>=1) 和异或 (x) 功能框程序段，可在其中为功能框输入和输出指定位值。 也可以连接到其它逻辑框并创建用户自己的逻辑组合。 在程序段中放置功能框后，可从“收藏夹”(Favorites) 工具栏或指令树中拖动“插入二进制输入”(Insert binary input) 工具，然后将其放置在功能框的输入侧以添加更多输入。 也可以右键单击功能框输入连接器并选择“插入输入”(Insert input)。

功能框输入和输出可连接到其它逻辑框，也可输入未连接输入的位地址或位符号名称。执行功能框指令时，当前输入状态会应用到二进制功能框逻辑，如果为真，功能框输出将为真。



AND 逻辑



OR 逻辑

XOR 逻辑

参数	数据类型	说明
IN1、IN2	Bool	输入位

- AND 功能框的所有输入必须都为“真”，输出才为“真”。
- OR 功能框只要有一个输入为“真”，输出就为“真”。
- XOR 功能框必须有奇数个输入为“真”，输出才为“真”。

NOT 逻辑反相器

对于 FBD 编程，可从“收藏夹”(Favorites) 工具栏或指令树中拖动“取反二进制输入”(Negate binary input) 工具，然后将其放置在输入或输出端以在该功能框连接器上创建逻辑反相器。

LAD： NOT 触点反相器

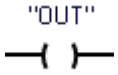
FBD： 带一个反向逻辑输入的 AND 功能框

FBD： 带反向逻辑输入和输出的 AND 功能框

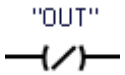
LAD NOT 触点取反能流输入的逻辑状态。

- 如果没有能流流入 NOT 触点，则会有能流流出。
- 如果有能流流入 NOT 触点，则没有能流流出。

LAD 输出线圈



输出线圈



反向输出线圈

线圈输出指令写入输出位的值。 如果用户指定的输出位使用存储器标识符 Q，则 CPU 接通或断开过程映像寄存器中的输出位，同时将指定的位设置为等于能流状态。 控制执行器的输出信号连接到 S7-1200 的 Q 端子。在 RUN 模式下，CPU 系统连续扫描输入信号，按照程序逻辑处理输入状态，然后通过在过程映像输出寄存器中设置新的输出状态值进行响应。 在每个程序执行循环之后，CPU 系统会将存储在过程映像寄存器中的新的输出状态响应传送到已连接的输出端子。

通过在 Q 偏移量后加入“:P”，可指定立即写入物理输出（例如：“%Q3.4:P”）。 对于立即写入，将位数据值写入过程映像输出并直接写入物理输出。

参数	数据类型	说明
OUT	Bool	分配位

- 如果有能流通过输出线圈，则输出位设置为 1。
- 如果没有能流通过输出线圈，则输出位设置为 0。
- 如果有能流通过反向输出线圈，则输出位设置为 0。
- 如果没有能流通过反向输出线圈，则输出位设置为 1。

FBD 输出分配功能框

在 FBD 编程中，LAD 线圈变为分配（= 和 /=）功能框，可在其中为功能框输出指定位地址。 功能框输入和输出可连接到其它功能框逻辑，用户也可以输入位地址。

输出分配

反向输出分配

带反向输出的输出分配

参数	数据类型	说明
OUT	Bool	分配位

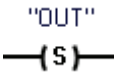
- 如果输出框输入为 1，则 OUT 位设置为 1。
- 如果输出框输入为 0，则 OUT 位设置为 0。
- 如果反向输出框输入为 1，则 OUT 位设置为 0。
- 如果反向输出框输入为 0，则 OUT 位设置为 1。

6.1.1.1 置位和复位指令

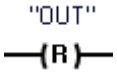
S 和 R： 置位和复位 1 位

- S（置位）激活时，OUT 地址处的数据值设置为 1。S 不激活时，OUT 不变。
- R（复位）激活时，OUT 地址处的数据值设置为 0。R 不激活时，OUT 不变。
- 这些指令可放置在程序段的任何位置。

LAD： 置位



LAD： 复位



FBD： 置位

FBD： 复位

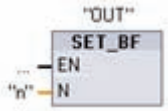
参数	数据类型	说明
IN（或连接到触点/门逻辑）	Bool	要监视的位位置
OUT	Bool	要置位或复位的位位置

SET_BF 和 RESET_BF： 置位和复位位域

LAD： SET_BF

LAD： RESET_BF

FBD： SET_BF

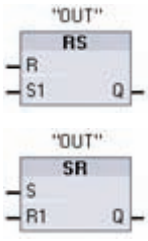


FBD：RESET_BF

参数	数据类型	说明
n	常数	要写入的位数
OUT	布尔数组的元素	要置位或复位的位域的起始元素 实例： #MyArray[3]

- SET_BF 激活时，为从地址 OUT 处开始的“n”位分配数据值 1。 SET_BF 不激活时，OUT 不变。
- RESET_BF 为从地址 OUT 处开始的“n”位写入数据值 0。 RESET_BF 不激活时，OUT 不变。
- 这些指令必须是分支中最右端的指令。

RS 和 SR： 置位优先和复位优先位锁存



RS 是置位优先锁存，其中置位优先。 如果置位 (S1) 和复位 (R) 信号都为真，则输出地址 OUT 将为 1。

SR 是复位优先锁存，其中复位优先。 如果置位 (S) 和复位 (R1) 信号都为真，则输出地址 OUT 将为 0。

OUT 参数指定置位或复位的位地址。 可选 OUT 输出 Q 反映“OUT”地址的信号状态。

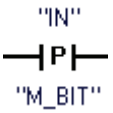
参数	数据类型	说明
S、S1	BOOL	置位输入；1 表示优先
R、R1	BOOL	复位输入；1 表示优先
OUT	BOOL	分配的位输出“OUT”
Q	BOOL	遵循“OUT”位的状态

指令	S1	R	“OUT”位
RS	0	0	先前状态
	0	1	0
	1	0	1
	1	1	1
	S	R1	
SR	0	0	先前状态
	0	1	0
	1	0	1
	1	1	0

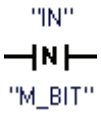
6.1.1.2 上升沿和下降沿指令

上升沿和下降沿跳变检测器

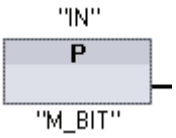
P 触点： LAD



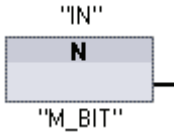
N 触点： LAD



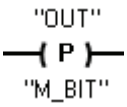
P 功能框： FBD



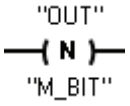
N 功能框： FBD



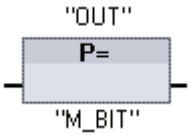
P 线圈： LAD



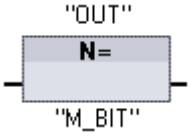
N 线圈： LAD



P= 功能框： FBD



N= 功能框： FBD



P_TRIG： LAD\FBD

N_TRIG： LAD\FBD

参数	数据类型	说明
M_BIT	Bool	保存输入的前一个状态的存储器位
IN	Bool	要检测其跳变沿的输入位
OUT	Bool	指示检测到跳变沿的输出位
CLK	Bool	要检测其跳变沿的能流或输入位
Q	Bool	指示检测到沿的输出

P 触点：
LAD
在分配的“IN”位上检测到正跳变（关到开）时，该触点的状态为 TRUE。该触点逻辑状态随后与能流输入状态组合以设置能流输出状态。 P 触点可以放置在程序段中除分支结尾外的任何位置。

N 触点：
LAD
在分配的输入位上检测到负跳变（开到关）时，该触点的状态为 TRUE。该触点逻辑状态随后与能流输入状态组合以设置能流输出状态。 N 触点可以放置在程序段中除分支结尾外的任何位置。

P 功能框：
FBD
在分配的输入位上检测到正跳变（关到开）时，输出逻辑状态为 TRUE。P 功能框只能放置在分支的开头。

N 功能框：
FBD
在分配的输入位上检测到负跳变（开到关）时，输出逻辑状态为 TRUE。N 功能框只能放置在分支的开头。

P 线圈：
LAD
在进入线圈的能流中检测到正跳变（关到开）时，分配的位“OUT”为 TRUE。 能流输入状态总是通过线圈后变为能流输出状态。 P 线圈可以放置在程序段中的任何位置。

N 线圈：
LAD
在进入线圈的能流中检测到负跳变（开到关）时，分配的位“OUT”为 TRUE。能流输入状态总是通过线圈后变为能流输出状态。 N 线圈可以放置在程序段中的任何位置。

P= 功能框：
FBD
在功能框输入连接的逻辑状态中或输入位赋值中（如果该功能框位于分支开头）检测到正跳变（关到开）时，分配的位“OUT”为 TRUE。 输入逻辑状态总是通过功能框后变为输出逻辑状态。 P= 功能框可以放置在分支中的任何位置。

N= 功能框：
FBD
在功能框输入连接的逻辑状态中或在输入位赋值中（如果该功能框位于分支开头）检测到负跳变（开到关）时，分配的位“OUT”为 TRUE。输入逻辑状态总是通过功能框后变为输出逻辑状态。 N= 功能框可以放置在分支中的任何位置。

P_TRIG： LAD/FBD	在 CLK 输入状态 (FBD) 或 CLK 能流输入 (LAD) 中检测到正跳变（关到开）时，Q 输出能流或逻辑状态为 TRUE。 在 LAD 中，P_TRIG 指令不能放置在程序段的开头或结尾。 在 FBD 中，P_TRIG 指令可以放置在除分支结尾外的任何位置。
N_TRIG (LAD/FBD)	在 CLK 输入状态 (FBD) 或 CLK 能流输入 (LAD) 中检测到负跳变（开到关）时，Q 输出能流或逻辑状态为 TRUE。 在 LAD 中，N_TRIG 指令不能放置在程序段的开头或结尾。 在 FBD 中，P_TRIG 指令可以放置在除分支结尾外的任何位置。

所有沿指令均使用存储器位 (M_BIT) 存储要监视的输入信号的前一个状态。 通过将输入的状态与存储器位的状态进行比较来检测沿。 如果状态指示在关注的方向上有输入变化，则会在输出写入 TRUE 来报告沿。 否则，输出会写入 FALSE。

说明

沿指令每次执行时都会对输入和存储器位值进行评估，包括第一次执行。 在程序设计期间必须考虑输入和存储器位的初始状态，以允许或避免在第一次扫描时进行沿检测。

由于存储器位必须从一次执行保留到下一次执行，所以应该对每个沿指令都使用唯一的位，并且不应在程序中的任何其它位置使用该位。 还应避免使用临时存储器和可受其它系统功能（例如 I/O 更新）影响的存储器。 仅将 M、全局 DB 或静态存储器（在背景 DB 中）用于 M_BIT 存储器分配。

6.1.2 定时器

使用定时器指令可创建编程的时间延迟：

- TP： 脉冲定时器可生成具有预设宽度时间的脉冲。
- TON： 接通延迟定时器输出 Q 在预设的延时过后设置为 ON。
- TOF： 关断延迟定时器输出 Q 在预设的延时过后重置为 OFF。
- TONR： 保持型接通延迟定时器输出在预设的延时过后设置为 ON。 在使用 R 输入重置经过的时间之前，会跨越多个定时时段一直累加经过的时间。
- RT： 通过清除存储在指定定时器背景数据块中的时间数据来重置定时器。

每个定时器都使用一个存储在数据块中的结构来保存定时器数据。 在编辑器中放置定时器指令时即可分配该数据块。

在功能块中放置定时器指令后，可以选择多重背景数据块选项，各数据结构的定时器结构名称可以不同，但定时器数据包含在单个数据块中，从而无需每个定时器都使用一个单独

的数据块。 这样可减少处理定时器所需的处理时间和数据存储空间。 在共享的多重背景数据块中的定时器数据结构之间不存在交互作用。

TP、TON 和 TOF 定时器具有相同的输入和输出参数。

TONR 定时器具有附加的复位输入参数 R。

可创建自己的“定时器名称”来命名定时器数据块，还可以描述该定时器在过程中的用途。

“定时器名称”
----[RT]----

RT 指令可重置指定定时器的定时器数据。

参数	数据类型	说明
IN	Bool	启用定时器输入
R	Bool	将 TONR 经过的时间重置为零
PT	Bool	预设的时间值输入
Q	Bool	定时器输出
ET	Time	经过的时间值输出
定时器数据块	DB	指定要使用 RT 指令复位的定时器

参数 IN 可启动和停止定时器：

- 参数 IN 从 0 跳变为 1 将启动定时器 TP、TON 和 TONR。
- 参数 IN 从 1 跳变为 0 将启动定时器 TOF。

下表列出了 PT 和 IN 参数值变化的影响。

定时器	PT 和 IN 参数值变化
TP	• 定时器运行期间，更改 PT 没有任何影响。 • 定时器运行期间，更改 IN 没有任何影响。
TON	• 定时器运行期间，更改 PT 没有任何影响。 • 定时器运行期间，将 IN 更改为 FALSE 会复位并停止定时器。

定时器	PT 和 IN 参数值变化
TOF	• 定时器运行期间，更改 PT 没有任何影响。 • 定时器运行期间，将 IN 更改为 TRUE 会复位并停止定时器。
TONR	• 定时器运行期间更改 PT 没有任何影响，但对定时器中断后继续运行会有影响。 • 定时器运行期间将 IN 更改为 FALSE 会停止定时器但不会复位定时器。将 IN 改回 TRUE 将使定时器从累积的时间值开始定时。

TIME 值

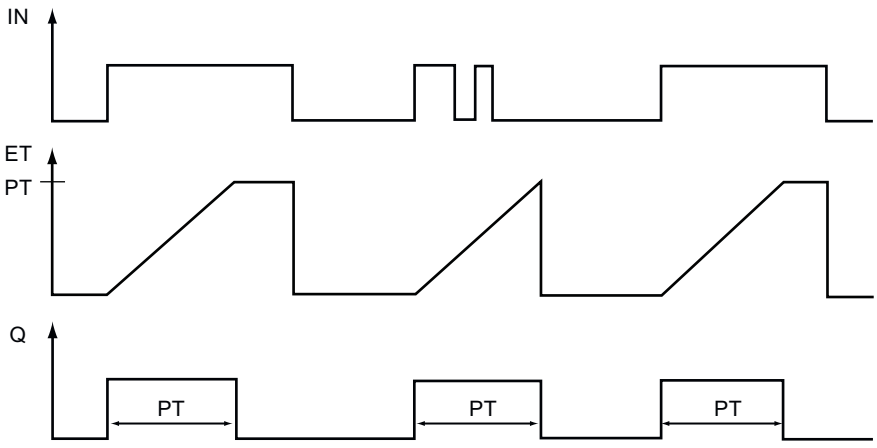
PT（预设时间）和 ET（经过的时间）值以表示毫秒时间的有符号双精度整数形式存储在存储器中。 TIME 数据使用 T# 标识符，可以简单时间单元“T#200ms”或复合时间单元“T#2s_200ms”的形式输入。

数据类型	大小	有效数值范围
TIME	32 位 存储形式	T#-24d_20h_31m_23s_648ms 到 T#24d_20h_31m_23s_647ms -2,147,483,648 ms 到 +2,147,483,647 ms

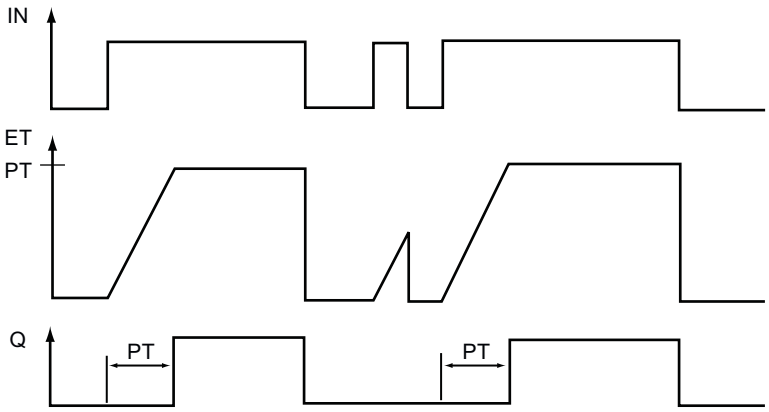
说明

在定时器指令中，无法使用上面所示 TIME 数据类型的负数范围。 负的 PT（预设时间）值在定时器指令执行时被设置为 0。 ET（经过的时间）始终为正值。

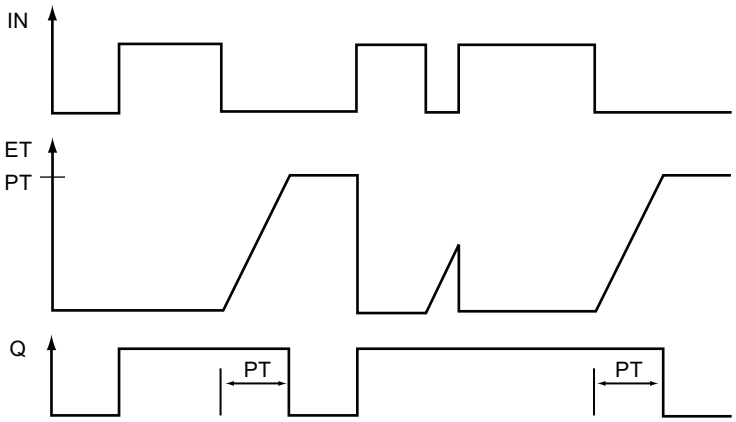
TP：
脉冲时序图



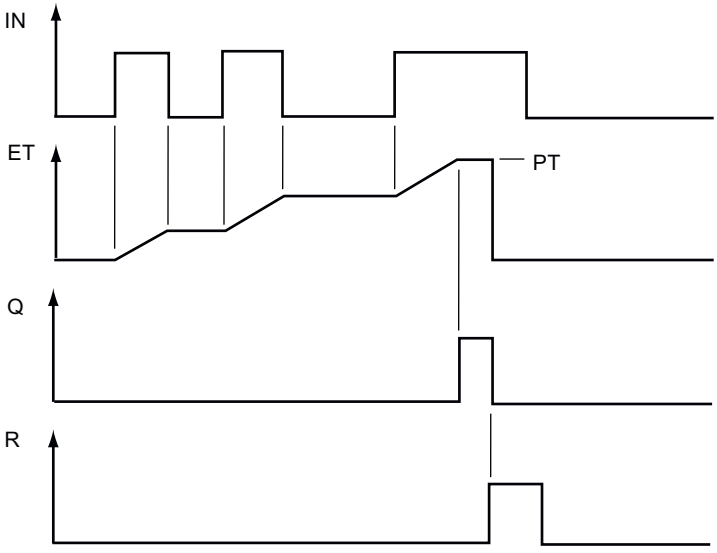
TON：接通延迟时序图



TOF：关断延迟时序图



TONR：保持型接通延迟时序图



6.1.3 计数器

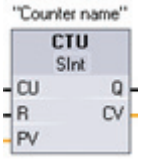
6.1.3.1 计数器

可使用计数器指令对内部程序事件和外部过程事件进行计数：

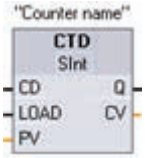
- CTU 是加计数器。
- CTD 是减计数器。
- CTUD 是加减计数器。

每个计数器都使用数据块中存储的结构来保存计数器数据。 用户在编辑器中放置计数器指令时分配相应的数据块。 这些指令使用软件计数器，软件计数器的最大计数速率受其所在的 OB 的执行速率限制。 指令所在的 OB 的执行频率必须足够高，以检测 CU 或 CD 输入的所有跳变。 要了解更快的计数操作，请参见 CTRL_HSC 指令。

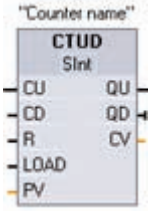
在功能块中放置计数器指令后，可以选择多重背景数据块选项，各数据结构的计数器结构名称可以不同，但计数器数据包含在单个数据块中，从而无需每个计数器都使用一个单独的数据块。 这减少了计数器所需的处理时间和数据存储空间。 在共享的多重背景数据块中的计数器数据结构之间不存在交互作用。



从功能框名称下的下拉列表中选择计数值数据类型。



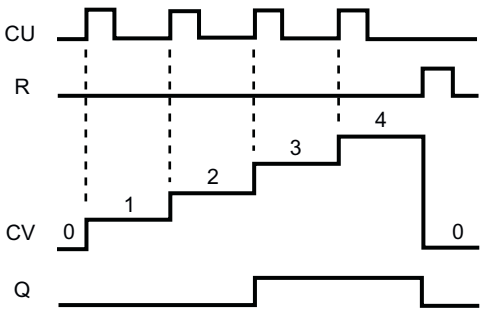
可创建自己的“计数器名称”来命名计数器数据块，还可以描述该计数器在过程中的用途。



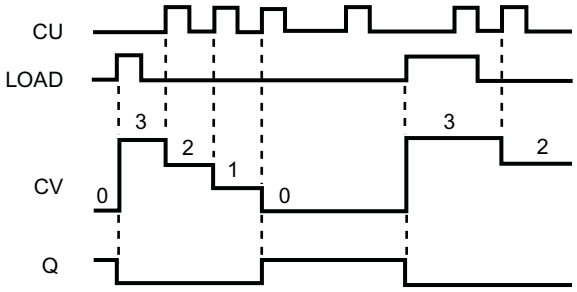
参数	数据类型	说明
CU、CD	Bool	加计数或减计数，按加或减一计数
R（CTU、CTUD）	Bool	将计数值重置为零
LOAD（CTD、CTUD）	Bool	预设值的装载控制
PV	SInt、Int、DInt、USInt、UInt、UDInt	预设计数值
Q、QU	Bool	CV >= PV 时为真
QD	Bool	CV <= 0 时为真
CV	SInt、Int、DInt、USInt、UInt、UDInt	当前计数值

计数值的数值范围取决于所选的数据类型。 如果计数值是无符号整型数，则可以减计数到零或加计数到范围限值。 如果计数值是有符号整数，则可以减计数到负整数限值或加计数到正整数限值。

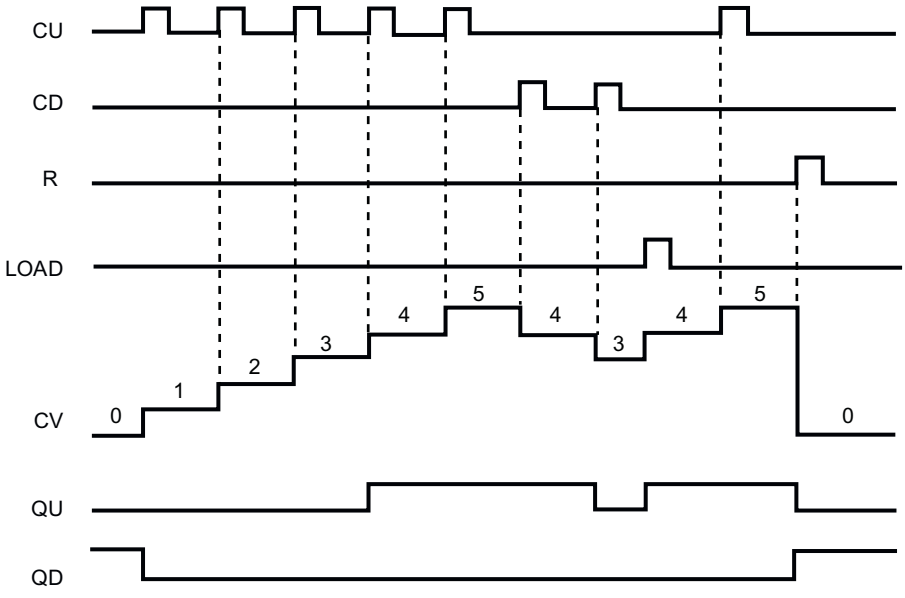
CTU： 参数 CU 的值从 0 变为 1 时，CTU 使计数值加 1。如果参数 CV（当前计数值）的值大于或等于参数 PV（预设计数值）的值，则计数器输出参数 Q = 1。如果复位参数 R 的值从 0 变为 1，则当前计数值复位为 0。下图显示了计数值是无符号整数时的 CTU 时序图（其中，PV = 3）。



CTD： 参数 CD 的值从 0 变为 1 时，CTD 使计数值减 1。如果参数 CV（当前计数值）的值等于或小于 0，则计数器输出参数 Q = 1。如果参数 LOAD 的值从 0 变为 1，则参数 PV（预设值）的值将作为新的 CV（当前计数值）装载到计数器。 下图显示了计数值是无符号整数时的 CTD 时序图（其中，PV = 3）。



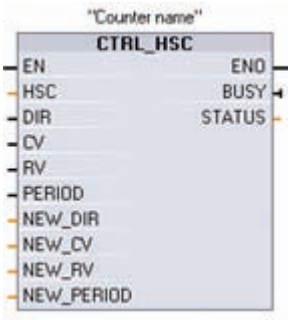
CTUD： 加计数 (CU, Count Up) 或减计数 (CD, Count Down) 输入的值从 0 跳变为 1 时，CTUD 会使计数值加 1 或减 1。 如果参数 CV（当前计数值）的值大于或等于参数 PV（预设值）的值，则计数器输出参数 QU = 1。如果参数 CV 的值小于或等于零，则计数器输出参数 QD = 1。如果参数 LOAD 的值从 0 变为 1，则参数 PV（预设值）的值将作为新的 CV（当前计数值）装载到计数器。 如果复位参数 R 的值从 0 变为 1，则当前计数值复位为 0。下图显示了计数值是无符号整数时的 CTUD 时序图（其中，PV = 4）。



6.1.3.2 CTRL_HSC 指令

CTRL_HSC 指令可控制高速计数器，这些高速计数器通常用来对发生速率比 OB 执行速率更快的事件进行计数。 CTU、CTD 和 CTUD 计数器指令的计数速率受其所在的 OB 的执行速率限制。 要了解 HSC 最大时钟输入频率，请参考 CPU 技术规范 (页 327)。

高速计数器的典型应用是对由运动控制轴编码器生成的脉冲进行计数。



每个 CTRL_HSC 指令都使用数据块中存储的结构来保存数据。 在编辑器中放置 CTRL_HSC 指令时即可分配该数据块。

可创建自己的“计数器名称”来命名计数器数据块，还可以描述该计数器在过程中的用途。

参数	参数类型	数据类型	说明
HSC	IN	HW_HSC	HSC 标识符
DIR	IN	Bool	1 = 请求新方向
CV	IN	Bool	1 = 请求设置新的计数器值
RV	IN	Bool	1 = 请求设置新的参考值
PERIOD	IN	Bool	1 = 请求设置新的周期值 （仅限频率测量模式）
NEW_DIR	IN	Int	新方向： 1= 向上 -1= 向下
NEW_CV	IN	DInt	新计数器值
NEW_RV	IN	DInt	新参考值
NEW_PERIOD	IN	Int	以秒为单位的新周期值： 0.01、0.1 或 1 （仅限频率测量模式）
BUSY	OUT	Bool	功能忙
STATUS	OUT	Word	执行条件代码

必须先在项目设置 PLC 设备配置中组态高速计数器，然后才能在程序中使用高速计数器。 HSC 设备配置设置包括选择计数模式、I/O 连接、中断分配以及是作为高速计数器还是设备来测量脉冲频率。 无论是否采用程序控制，均可操作高速计数器。

许多高速计数器组态参数只在项目设备配置中进行设置。 有些高速计数器参数在项目设备配置中初始化，但以后可以通过程序控制进行修改。

CTRL_HSC 指令参数提供了计数过程的程序控制：

- 将计数方向设置为 NEW_DIR 值
- 将当前计数值设置为 NEW_CV 值
- 将参考值设置为 NEW_RV 值
- 将周期值（仅限频率测量模式）设置为 NEW_PERIOD 值

如果执行 CTRL_HSC 指令后以下布尔标记值置位为 1，则相应的 NEW_xxx 值将装载到计数器。 CTRL_HSC 指令执行一次可处理多个请求（同时设置多个标记）。

- DIR = 1 是装载 NEW_DIR 值的请求，0 = 无变化
- CV = 1 是装载 NEW_CV 值的请求，0 = 无变化
- RV = 1 是装载 NEW_RV 值的请求，0 = 无变化
- PERIOD = 1 是装载 NEW_PERIOD 值的请求，0 = 无变化

CTRL_HSC 指令通常放置在触发计数器硬件中断事件时执行的硬件中断 OB 中。 例如，如果 CV=RV 事件触发计数器中断，则硬件中断 OB 代码块执行 CTRL_HSC 指令并且可通过装载 NEW_RV 值更改参考值。

在 CTRL_HSC 参数中没有提供当前计数值。 在高速计数器硬件配置期间分配存储当前计数值的过程映像地址。 可以使用程序逻辑直接读取该计数值并且返回到程序的值将是读取计数器瞬间的正确计数。 但计数器仍将继续对高速事件计数。 因此，程序使用旧的计数值完成处理前，实际计数值可能会更改。

CTRL_HSC 参数的详细信息：

- 如果不请求更新参数值，则会忽略相应的输入值。
- 仅当组态的计数方向设置为“用户程序（内部方向控制）”(User program (internal direction control)) 时，DIR 参数才有效。 用户在 HSC 设备配置中确定如何使用该参数。
- 对于 CPU 或信号板上的 S7-1200 HSC，BUSY 参数的值始终为 0。

条件代码： 发生错误时，ENO 设置为 0，并且 STATUS 输出包含条件代码。

STATUS 值 (W#16#...)	说明
0	无错误
80A1	HSC 标识符没有对 HSC 寻址
80B1	NEW_DIR 的值非法
80B2	NEW_CV 的值非法
80B3	NEW_RV 的值非法
80B4	NEW_PERIOD 的值非法

6.1.3.3 高速计数器的使用方法

高速计数器 (HSC) 可用作增量轴编码器的输入。 该轴编码器每转提供指定数量的计数值以及一个复位脉冲。 来自轴编码器的时钟和复位脉冲将输入到 HSC 中。

先是将若干预设值中的第一个装载到 HSC 上，并且在当前计数值小于当前预设值的时段内计数器输出一直是激活的。 在当前计数值等于预设时、发生复位时以及方向改变时，HSC 会提供一个中断。

每次出现“当前计数值等于预设值”中断事件时，将装载一个新的预设值，同时设置输出的下一状态。 当出现复位中断事件时，将设置输出的第一个预设值和第一个输出状态，并重复该循环。

由于中断发生的频率远低于 HSC 的计数速率，因此能够在对 CPU 扫描周期影响相对较小的情况下实现对高速操作的精确控制。 通过提供中断，可以在独立的中断例程中执行每次的新预设值装载操作以实现简单的状态控制。 （或者，所有中断事件也可在单个中断例程中进行处理。）

选择 HSC 的功能

所有 HSC 在同种计数器运行模式下的工作方式都相同。 HSC 共有四种基本类型：

- 具有内部方向控制的单相计数器
- 具有外部方向控制的单相计数器
- 具有 2 个时钟输入的双相计数器
- A/B 相正交计数器

用户可选择是否激活复位输入来使用各种 HSC 类型。 如果激活复位输入（存在一些限制，请参见下表），则它会清除当前值并在您禁用复位输入之前保持清除状态。

- 频率功能： 有些 HSC 模式允许 HSC 被组态（计数类型）为报告频率而非当前脉冲计数值。 有三种可用的频率测量周期： 0.01、0.1 或 1.0 秒。

  频率测量周期决定 HSC 计算并报告新频率值的频率。 报告频率是通过上一测量周期内总计数值确定的平均值。 如果该频率在快速变化，则报告值将是介于测量周期内出现的最高频率和最低频率之间的一个中间值。 无论频率测量周期的设置是什么，总是会以赫兹为单位来报告频率（每秒脉冲个数）。

- 计数器模式和输入： 下表列出了用于与 HSC 相关的时钟、方向控制和复位功能的输入。

  同一输入不可用于两个不同的功能，但任何未被其 HSC 的当前模式使用的输入均可用于其它用途。 例如，如果 HSC1 处于使用内置输入但不使用外部复位 (I0.3) 的模式，则 I0.3 可以用于沿中断或 HSC2。

说明			默认输入分配			功能
HSC	HSC1	内置 或信号板 或监视 PTO 0[1]	I0.0 I4.0 PTO 0 脉冲	I0.1 I4.1 PTO 0 方向	I0.3 I4.3 -	
	HSC：	内置 或信号板 或监视 PTO 1[1]	I0.2 I4.2 PTO 1 脉冲	I0.3 I4.3 PTO 1 方向	I0.1 I4.1 -	
	HSC3[2]	内置	I0.4	I0.5	I0.7	
	HSC4[3]	内置	I0.6	I0.7	I0.5	
	HSC5[4]	内置 或信号板	I1.0 I4.0	I1.1 I4.1	I1.2 I4.3	
	HSC6[4]	内置 或信号板	I1.3 I4.2	I1.4 I4.3	I1.5 I4.1	
模式	具有内部方向控制的单相计数器		时钟	-	-	计数或频率
					复位	计数
	具有外部方向控制的单相计数器		时钟	方向	-	计数或频率
					复位	计数
	具有 2 个时钟输入的双相计数器		加时钟	减时钟	-	计数或频率
					复位	计数
	A/B 相正交计数器		A 相	B 相	-	计数或频率
					Z 相	计数
	监视脉冲串输出 (PTO)[1]		时钟	方向	-	计数

1 脉冲串输出监视功能始终使用时钟和方向。 如果仅为脉冲组态了相应的 PTO 输出，则通常应将方向输出设置为正计数。

2 对于仅支持 6 个内置输入的 CPU 1211C，不能使用带复位输入的 HSC3。

3 对于仅支持 6 个内置输入的 CPU 1211C，不能使用 HSC4。

4 仅当安装信号板时，CPU 1211C 和 CPU 1212C 才支持 HSC5 和 HSC6。

访问 HSC 的当前值

CPU 将每个 HSC 的当前值存储在一个输入 (I) 地址中。 下表列出了为每个 HSC 的当前值分配的默认地址。 可以通过在设备配置中修改 CPU 的属性来更改当前值的 I 地址。

高速计数器	数据类型	默认地址
HSC1	DInt	ID1000
HSC2	DInt	ID1004
HSC3	DInt	ID1008
HSC4	DInt	ID1012
HSC5	DInt	ID1016
HSC6	DInt	ID1020

无法强制分配给 HSC 设备的数字量 I/O 点

在设备配置期间分配高速计数器设备使用的数字量 I/O 点。将数字 I/O 点分配给这些设备之后，无法通过监视表格强制功能修改所分配的 I/O 点的地址值。

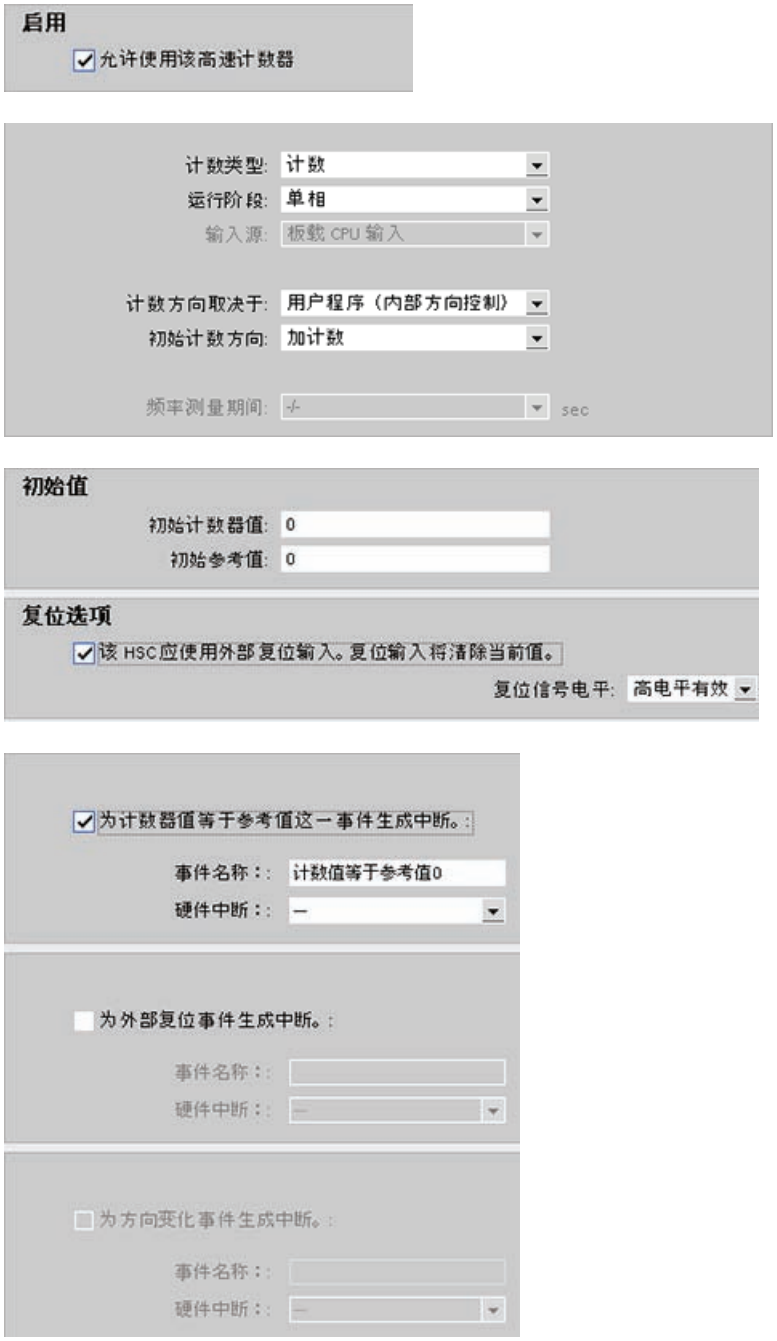
6.1.3.4 组态 HSC

CPU 允许用户组态最多 6 个高速计数器。 用户可编辑 CPU 的“属性”(Properties) 来组态各个 HSC 的参数。

通过编辑 CPU 的“属性”(Properties) 来组态高速计数器的参数。

启用 HSC 之后组态其它参数，例如计数器功能、初始值、复位选项和中断事件。

组态 HSC 之后，在用户程序中使用 CTRL_HSC 指令控制 HSC 的运行。



6.1.4 比较

LAD FBD

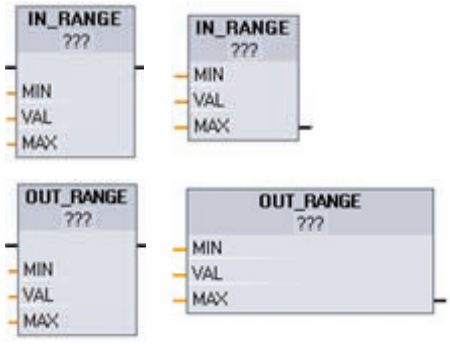
使用比较指令可比较两个数据类型相同的值。 该 LAD 触点比较结果为 TRUE 时，则该触点会被激活。 如果该 FBD 功能框比较结果为 TRUE，则功能框输出为 TRUE。

在程序编辑器中单击该指令后，可以从下拉菜单中选择比较类型和数据类型。

关系类型	满足以下条件时比较结果为真：
==	IN1 等于 IN2
<>	IN1 不等于 IN2
>=	IN1 大于或等于 IN2
<=	IN1 小于或等于 IN2
>	IN1 大于 IN2
<	IN1 小于 IN2

参数	数据类型	说明
IN1, IN2	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, String, Char, Time, DTL, Constant	要比较的值

范围内和范围外指令



LAD FBD

使用 IN_RANGE 和 OUT_RANGE 指令可测试输入值是在指定的值范围之内还是之外。 如果比较结果为 TRUE，则功能框输出为 TRUE。

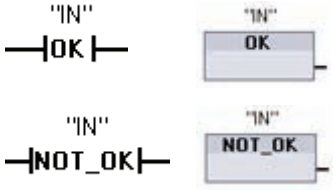
输入参数 MIN、VAL 和 MAX 的数据类型必须相同。

在程序编辑器中单击该指令后，可以从下拉菜单中选择数据类型。

关系类型	满足以下条件时比较结果为 TRUE：
IN_RANGE	MIN <= VAL <= MAX
OUT_RANGE	VAL < MIN 或 VAL > MAX

参数	数据类型	说明
MIN、VAL、MAX	SInt、Int、DInt、USInt、UInt、UDInt、Real、Constant	比较器输入

OK 和 Not OK 指令



LAD FBD

使用 OK 和 NOT_OK 指令可测试输入的参考数据是否为符合 IEEE 规范 754 的有效实数。如果该 LAD 触点为 TRUE，则激活该触点并传递能流。 如果该 FBD 功能框为 TRUE，则功能框输出为 TRUE。

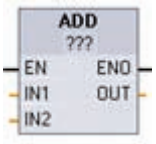
如果 Real 或 LReal 类型的值为 +/- INF（无穷大）、NaN（不是数字）或者非标准化的值，则其无效。 非标准化的值是非常接近于 0 的数字。 CPU 在计算中用 0 替换非标准化的值。

指令	满足以下条件时 REAL 数测试结果为 TRUE：
OK	输入值为有效 REAL 数
NOT_OK	输入值不是有效 REAL 数

参数	数据类型	说明
IN	Real, LReal	输入数据

6.1.5 数学

加法、减法、乘法和除法指令



使用数学功能框指令可编写基本数学运算程序：

- ADD： 加法 (IN1 + IN2 = OUT)
- SUB： 减法 (IN1 - IN2 = OUT)
- MUL： 乘法 (IN1 * IN2 = OUT)
- DIV： 除法 (IN1 / IN2 = OUT)

  整数除法运算会截去商的小数部分以生成整数输出。

在功能框名称下方单击，并从下拉菜单中选择数据类型。

说明

基本数学指令参数 IN1、IN2 和 OUT 的数据类型必须相同。

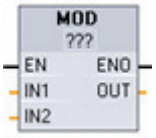
参数	数据类型	说明
IN1, IN2	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Constant	数学运算输入
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	数学运算输出

启用数学指令 (EN = 1) 后，指令会对输入值（IN1 和 IN2）执行指定的运算并将结果存储在通过输出参数 (OUT) 指定的存储器地址中。 运算成功完成后，指令会设置 ENO = 1。

ENO 状态	说明
1	无错误
0	数学运算结果值可能超出所选数据类型的有效数值范围。 返回适合目标大小的结果的最低有效部分。
0	除数为 0 (IN2 = 0)： 结果未定义，返回 0。
0	Real/LReal： 如果其中一个输入值为 NaN（不是数字），则返回 NaN。
0	ADD Real/LReal： 如果两个 IN 值均为 INF，但符号不同，则这是非法运算并返回 NaN。

ENO 状态	说明
0	SUB Real/LReal： 如果两个 IN 值均为 INF，且符号相同，则这是非法运算并返回 NaN。
0	MUL Real/LReal： 如果一个 IN 值为零而另一个为 INF，则这是非法运算并返回 NaN。
0	DIV Real/LReal： 如果两个 IN 值均为零或 INF，则这是非法运算并返回 NaN。

6.1.5.1 MOD 指令



MOD（求模）指令用于 IN1 以 IN2 为模的数学运算。 运算 IN1 MOD IN2 = IN1 - (IN1 / IN2) = 参数 OUT。

在功能框名称下方单击，并从下拉菜单中选择数据类型。

说明

IN1、IN2 和 OUT 参数的数据类型必须相同。

参数	数据类型	说明
IN1 和 IN2	Int、DInt、USInt、UInt、UDInt、Constant	求模输入
OUT	Int、DInt、USInt、UInt、UDInt	求模输出

ENO 状态	说明
1	无错误
0	值 IN2 = 0，OUT 值为零

NEG 指令

使用 NEG（取反）指令可将参数 IN 的值的算术符号取反并将结果存储在参数 OUT 中。

在功能框名称下方单击，并从下拉菜单中选择数据类型。

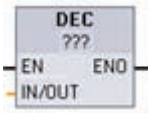
说明

IN 和 OUT 参数的数据类型必须相同。

参数	数据类型	说明
IN	SInt, Int, DInt, Real, LReal, Constant	数学运算输入
OUT	SInt, Int, DInt, Real, LReal	数学运算输出

ENO 状态	说明
1	无错误
0	结果值超出所选数据类型的有效数值范围。 以 SInt 为例： NEG (-128) 的结果为 +128，超出该数据类型的最大值。

递增和递减指令



INC 和 DEC 指令用于：

- 递增有符号或无符号整数值

  INC（递增）： 参数 IN/OUT 值 +1 = 参数 IN/OUT 值
- 递减有符号或无符号整数值

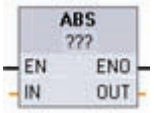
  DEC（递减）： 参数 IN/OUT 值 - 1 = 参数 IN/OUT 值

在功能框名称下方单击，并从下拉菜单中选择数据类型。

参数	数据类型	说明
IN/OUT	SInt, Int, DInt, USInt, UInt, UDInt	数学运算输入和输出

ENO 状态	说明
1	无错误
0	结果值超出所选数据类型的有效数值范围。 以 SInt 为例： INC (127) 的结果为 -128，超出该数据类型最大值。

绝对值指令



使用 ABS 指令可以对参数 IN 的有符号整数或实数求绝对值并将结果存储在参数 OUT 中。

在功能框名称下方单击，并从下拉菜单中选择数据类型。

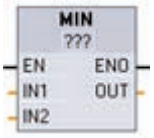
说明

IN 和 OUT 参数的数据类型必须相同。

参数	数据类型	说明
IN	SInt, Int, DInt, Real, LReal	数学运算输入
OUT	SInt, Int, DInt, Real, LReal	数学运算输出

ENO 状态	说明
1	无错误
0	数学运算结果值超出所选数据类型的有效数值范围。 以 SInt 为例： ABS (-128) 的结果为 +128，超出该数据类型最大值。

MIN 和 MAX 指令



按如下说明使用 MIN（最小值）和 MAX（最大值）指令：

- MIN 比较两个参数 IN1 和 IN2 的值并将最小（较小）值分配给参数 OUT。
- MAX 比较两个参数 IN1 和 IN2 的值并将最大（较大）值分配给参数 OUT。

在功能框名称下方单击，并从下拉菜单中选择数据类型。

说明

IN1、IN2 和 OUT 参数的数据类型必须相同。

参数	数据类型	说明
IN1, IN2	SInt, Int, DInt, USInt, UInt, UDInt, Real, Constant	数学运算输入
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real	数学运算输出

ENO 状态	说明
1	无错误
0	仅适用于 Real 数据类型： • 一个或两个输入不是 REAL 数 (NaN)。 • 结果 OUT 为 +/- INF（无穷大）。

Limit 指令

使用 Limit 指令测试参数 IN 的值是否在参数 MIN 和 MAX 指定的值范围内。如果 IN 值超出该范围，OUT 值将固定为 MIN 或 MAX 值。

- 如果参数 IN 的值在指定的范围内，则 IN 的值将存储在参数 OUT 中。
- 如果参数 IN 的值超出指定的范围，则 OUT 值为参数 MIN 的值（如果 IN 值小于 MIN 值）或参数 MAX 的值（如果 IN 值大于 MAX 值）。

在功能框名称下方单击，并从下拉菜单中选择数据类型。

说明

MIN、IN、MAX 和 OUT 参数的数据类型必须相同。

参数	数据类型	说明
MIN, IN 和 MAX	SInt, Int, DInt, USInt, UInt, UDInt, Real, Constant	数学运算输入
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real	数学运算输出

ENO 状态	说明
1	无错误
0	Real： 如果 MIN、IN 和 MAX 的一个或多个值是 NaN（不是数字），则返回 NaN。
0	如果 MIN 大于 MAX，则将值 IN 分配给 OUT。

浮点型算术运算指令

使用浮点指令可编写使用 Real 或 LReal 数据类型的数学运算程序：

- SQR： 平方 ($IN^2 = OUT$)
- SQRT： 平方根 ($\sqrt{IN} = OUT$)
- LN： 自然对数 (LN(IN) = OUT)

- EXP： 自然指数 (e^{IN}=OUT)，其中底数 e = 2.71828182845904523536
- SIN： 正弦 (sin(IN 弧度) = OUT)
- COS： 余弦 (cos(IN 弧度) = OUT)
- TAN： 正切 (tan(IN 弧度) = OUT)
- ASIN： 反正弦 (arcsine(IN) = OUT 弧度)，其中 sin(OUT 弧度) = IN
- ACOS： 反余弦 (arccos(IN) = OUT 弧度)，其中 cos(OUT 弧度) = IN
- ATAN： 反正切 (arctan(IN) = OUT 弧度)，其中 tan(OUT 弧度) = IN
- FRAC： 分数（浮点数 IN 的小数部分 = OUT）
- EXPT： 一般指数 ($IN1^{IN2}$ = OUT)

在功能框名称下方单击，并从下拉菜单中选择数据类型。 EXPT 参数 IN1 和 OUT 始终为实数。 可以选择指数参数 IN2 的数据类型。

参数	数据类型	说明
IN, IN1	Real, LReal, Constant	输入
IN2	SInt, Int, DInt, USInt, UInt,UDInt, Real, LReal, Constant	EXPT 指数输入
OUT	Real, LReal	输出

ENO 状态	指令	条件	结果 (OUT)
1	全部	无错误	有效结果
0	SQR	结果超出有效 Real/LReal 范围	+INF
		IN 为 +/- NaN（不是数字）	+NaN
	SQRT	IN 为负数	-NaN
		IN 为 +/- INF（无穷大）或 +/- NaN	+/- INF 或 +/- NaN
	LN	IN 为 0.0、负数、-INF 或 -NaN	-NaN
		IN 为 +INF 或 +NaN	+INF 或 +NaN
	EXP	结果超出有效 Real/LReal 范围	+INF
		IN 为 +/- NaN	+/- NaN
	SIN、COS、TAN	IN 为 +/- INF 或 +/- NaN	+/- INF 或 +/- NaN
	ASIN、ACOS	IN 超出 -1.0 到 +1.0 的有效范围	+NaN
		IN 为 +/- NaN	+/- NaN
	ATAN	IN 为 +/- NaN	+/- NaN
	FRAC	IN 为 +/- INF 或 +/- NaN	+NaN
	EXPT	IN1 为 +INF 且 IN2 不是 -INF	+INF
		IN1 为负数或 -INF	如果 IN2 为 Real/LReal，则为 +NaN， 否则为 -INF
		IN1 或 IN2 为 +/- NaN	+NaN
		IN1 为 0.0 且 IN2 为 Real/LReal（只能为 Real/LReal）	+NaN

6.1.6 移动

移动和块移动指令

使用移动指令将数据元素复制到新的存储器地址并从一种数据类型转换为另一种数据类型。 移动过程不会更改源数据。

- MOVE： 将存储在指定地址的数据元素复制到新地址
- MOVE_BLK： 将数据元素块复制到新地址的可中断移动
- UMOVE_BLK： 将数据元素块复制到新地址的不中断移动

MOVE		
参数	**数据类型**	**说明**
IN	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Char, Array, Struct, DTL, Time	源地址
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Char, Array, Struct, DTL, Time	目标地址

MOVE_BLK、UMOVE_BLK		
参数	**数据类型**	**说明**
IN	SInt, Int, DInt, USInt, UInt, UDInt, Real, Byte, Word, DWord	源起始地址
COUNT	UInt	要复制的数据元素数
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, Byte, Word, DWord	目标起始地址

说明

数据复制操作规则

- 要复制 Bool 数据类型，请使用 SET_BF、RESET_BF、R、S 或输出线圈 (LAD)
- 要复制单个基本数据类型，请使用 MOVE
- 要复制基本数据类型数组，请使用 MOVE_BLK 或 UMOVE_BLK
- 要复制结构，请使用 MOVE
- 要复制字符串，请使用 S_CONV
- 要复制字符串中的单个字符，请使用 MOVE
- MOVE_BLK 和 UMOVE_BLK 指令不能用于将数组或结构复制到 I、Q 或 M 存储区。

MOVE 指令将单个数据元素从 IN 参数指定的源地址复制到 OUT 参数指定的目标地址。

MOVE_BLK 和 UMOVE_BLK 指令具有附加的 COUNT 参数。 COUNT 指定要复制的数据元素个数。 每个被复制元素的字节数取决于 PLC 变量表中分配给 IN 和 OUT 参数变量名称的数据类型。

MOVE_BLK 和 UMOVE_BLK 指令在处理中断的方式上有所不同：

- 在 MOVE_BLK 执行期间**排队并处理**中断事件。 在中断 OB 子程序中未使用移动目标地址的数据时，或者虽然使用了该数据，但目标数据不必一致时，使用 MOVE_BLK 指令。 如果 MOVE_BLK 操作被中断，则最后移动的一个数据元素在目标地址中是完整并且一致的。 MOVE_BLK 操作会在中断 OB 执行完成后继续执行。
- 在 UMOVE_BLK 完成执行前**排队但不处理**中断事件。 如果在执行中断 OB 子程序前移动操作必须完成且目标数据必须一致，则使用 UMOVE_BLK 指令。 更多信息，请参阅数据一致性 (页 94)部分。

执行 MOVE 指令之后，ENO 始终为真。

ENO 状态	条件	结果
1	无错误	成功复制了全部的 COUNT 个元素
0	源 (IN) 范围或目标 (OUT) 范围超出可用存储区	复制适当的元素。 不复制部分元素。

填充指令

按如下说明使用 FILL_BLK 和 UFILL_BLK 指令：

- FILL_BLK： 可中断填充指令使用指定数据元素的副本填充地址范围。
- UFILL_BLK： 不中断填充指令使用指定数据元素的副本填充地址范围。

参数	数据类型	说明
IN	SInt, Int, DIntT, USInt, UInt, UDInt, Real, BYTE, Word, DWord	数据源地址
COUNT	USInt, UInt	要复制的数据元素数
OUT	SInt, Int, DIntT, USInt, UInt, UDInt, Real, BYTE, Word, DWord	数据目标地址

说明

数据填充操作规则

- 要使用 BOOL 数据类型填充，请使用 SET_BF、RESET_BF、R、S 或输出线圈 (LAD)
- 要使用单个基本数据类型填充，请使用 MOVE
- 要使用基本数据类型填充数组，请使用 FILL_BLK 或 UFILL_BLK
- 要在字符串中填充单个字符，请使用 MOVE
- FILL_BLK 和 UFILL_BLK 指令不能用于将数组填充到 I、Q 或 M 存储区。

FILL_BLK 和 UFILL_BLK 指令将源数据元素 IN 复制到通过参数 OUT 指定其初始地址的目标中。 复制过程不断重复并填充相邻地址块，直到副本数等于 COUNT 参数。

FILL_BLK 和 UFILL_BLK 指令在处理中断的方式上有所不同：

- 在 FILL_BLK 执行期间**排队并处理**中断事件。 在中断 OB 子程序中未使用移动目标地址的数据时，或者虽然使用了该数据，但目标数据不必一致时，使用 FILL_BLK 指令。
- 在 UFILL_BLK 完成执行前**排队但不处理**中断事件。 如果在执行中断 OB 子程序前移动操作必须完成且目标数据必须一致，则使用 UFILL_BLK 指令。

ENO 状态	条件	结果
1	无错误	IN 元素成功复制到全部的 COUNT 个目标中。
0	目标 (OUT) 范围超出可用存储区	复制适当的元素。不复制部分元素。

6.1.6.1 交换指令

SWAP 指令用于调换二字节和四字节数据元素的字节顺序。 不改变每个字节中的位顺序。 执行 SWAP 指令之后，ENO 始终为 TRUE。

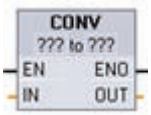
在功能框名称下方单击，并从下拉菜单中选择数据类型。

参数	数据类型	说明
IN	Word, DWord	有序数据字节 IN
OUT	Word, DWord	反转有序数据字节 OUT

	示例： 参数 IN = MB0 SWAP 执行前				实例： 参数 OUT = MB4， SWAP 执行后			
地址	MB0	MB 1			MB4	MB 5		
W#16#1234	12	34			34	12		
WORD	MSB	LSB			MSB	LSB		
地址	MB0	MB 1	MB 2	MB3	MB4	MB 5	MB 6	MB7
DW#16#12345678	12	34	56	78	78	56	34	12
DWORD	MSB			LSB	MSB			LSB

6.1.7 转换

转换指令



CONVERT 指令用于将数据元素从一种数据类型转换为另一种数据类型。 在功能框名称下方单击，然后从下拉列表中选择 IN 数据类型和 OUT 数据类型。

选择（转换源）数据类型之后，（转换目标）下拉列表中将显示可能的转换项列表。 与 BCD16 进行相互转换仅限于 Int 数据类型。与 BCD32 进行转换仅限于 DInt 数据类型。

在功能框名称下方单击，并从下拉菜单中选择数据类型。

参数	数据类型	说明
IN	SInt, Int, DInt, USInt, UInt, UDInt, Byte, Word, DWord, Real, LReal, Bcd16, Bcd32	IN 值
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Byte, Word, DWord, Real, LReal, Bcd16, Bcd32	转换为新数据类型的 IN 值

ENO 状态	说明	结果 (OUT)
1	无错误	有效结果
0	IN 为 +/- INF 或 +/- NaN	+/- INF 或 +/- NaN
0	结果超出 OUT 数据类型的有效范围	OUT 被设置为 IN 的最低有效字节

取整和截取指令

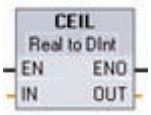
ROUND 用于将实数转换为整数。 实数的小数部分舍入为最接近的整数值（IEEE - 舍入为最接近值）。 如果 Real 数刚好是两个连续整数的一半（例如，10.5），则 Real 数舍入为偶数。 例如，ROUND (10.5) = 10 或 ROUND (11.5) = 12。

TRUNC 用于将实数转换为整数。 实数的小数部分被截成零（IEEE - 取整为零）。

参数	数据类型	说明
IN	Real, LReal	浮点型输入
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	取整或截取后的输出

ENO 状态	说明	结果 (OUT)
1	无错误	有效结果
0	IN 为 +/- INF 或 +/- NaN	+/- INF 或 +/- NaN

上取整和下取整指令



CEIL 用于将实数转换为大于或等于该实数的最小整数（IEEE - 向正无穷取整）。

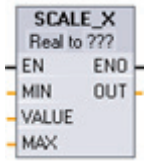
FLOOR 用于将实数转换为小于或等于该实数的最大整数（IEEE - 向负无穷取整）。

参数	数据类型	说明
IN	Real, LReal	浮点型输入
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	转换后的输出

ENO 状态	说明	结果 (OUT)
1	无错误	有效结果
0	IN 为 +/- INF 或 +/- NaN	+/- INF 或 +/- NaN

6.1.7.1 标定和标准化指令

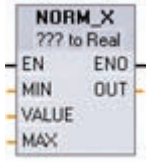
标定和标准化指令



SCALE_X 用于按参数 MIN 和 MAX 所指定的数据类型和值范围对标准化的实参数 VALUE（其中，0.0 <= VALUE <= 1.0）进行标定：

OUT = VALUE (MAX - MIN) + MIN

对于 SCALE_X，参数 MIN、MAX 和 OUT 的数据类型必须相同。



NORM_X 用于标准化通过参数 MIN 和 MAX 指定的值范围内的参数 VALUE：

OUT = (VALUE - MIN) / (MAX - MIN)，其中，0.0 <= OUT <= 1.0

对于 NORM_X，参数 MIN、VALUE 和 MAX 的数据类型必须相同。

在功能框名称下方单击，并从下拉菜单中选择数据类型。

参数	数据类型	说明
MIN	SInt、Int、DInt、USInt、UInt、UDInt、Real	输入范围的最小值
VALUE	SCALE_X：Real NORM_X： SInt、Int、DInt、USInt、UInt、UDInt、Real	要标定或标准化的输入值

参数	数据类型	说明
MAX	SInt、Int、DInt、USInt、UInt、UDInt、Real	输入范围的最大值
OUT	SCALE_X： SInt、Int、DInt、USInt、UInt、UDInt、Real NORM_X：Real	标定或标准化后的输出值

说明

SCALE_X 的参数 VALUE 应限制为 (0.0 <= VALUE <= 1.0)

如果参数 VALUE 小于 0.0 或大于 1.0：

- 线性标定运算会生成一些小于参数 MIN 值或大于参数 MAX 值的 OUT 值，作为落在 OUT 数据类型值范围内的 OUT 值。 此时，SCALE_X 执行会设置 ENO = TRUE。
- 可能会生成不在 OUT 数据类型范围内的一些标定数。 此时，参数 OUT 的值会被设置为一个中间值，该中间值等于被标定实数在最终转换为 OUT 数据类型之前的最低有效部分。 此时，SCALE_X 执行会设置 ENO = FALSE。

NORM_X 的参数 VALUE 应限制为 (MIN <= VALUE <= MAX)

如果参数 VALUE 小于 MIN 或大于 MAX，线性标定运算会生成小于 0.0 或大于 1.0 的标准化 OUT 值。 在这种情况下，NORM_X 执行会设置 ENO = TRUE。

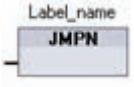
ENO 状态	条件	结果 (OUT)
1	无错误	有效结果
0	结果超出 OUT 数据类型的有效范围	中间结果： 实数在最终转换为 OUT 数据类型前的最低有效部分。
0	参数 MAX <= MIN	SCALE_X： 用 Real 数 VALUE 的最低有效部分填充 OUT 大小。 NORM_X： 扩展 VALUE 数据类型的 VALUE 来填充双字大小。
0	参数 VALUE = +/- INF 或 +/- NaN	将 VALUE 写入 OUT

6.1.8 程序控制

跳转和标签指令

程序控制指令用于有条件地控制执行顺序：

JMP： 如果有能流通过 JMP 线圈 (LAD)，或者 JMP 功能框的输入为真 (FBD)，则程序将从指定标签后的第一条指令继续执行。



JMPN： 如果没有能流通过 JMP 线圈 (LAD)，或者 JMP 功能框的输入为假 (FBD)，则程序将从指定标签后的第一条指令继续执行。

标签： JMP 或 JMPN 跳转指令的目标标签。

LAD **FBD**

参数	数据类型	说明
Label_name	标签标识符	跳转指令以及相应跳转目标程序标签的标识符

通过在 LABEL 指令中直接键入来创建标签名称。 可以使用参数助手图标来选择 JMP 和 JMPN 标签名称域可用的标签名称。 也可在 JMP 或 JMPN 指令中直接键入标签名称。

Return_Value (RET) 执行控制指令

LAD **FBD**

RET 指令用于终止当前块的执行。

参数	数据类型	说明
Return_Value	Bool	RET 指令的“Return_value”参数被分配给调用块中块调用功能框的 ENO 输出。

可选的 RET 指令用于终止当前块的执行。 当且仅当有能流通过 RET 线圈 (LAD)，或者当 RET 功能框的输入为真 (FBD) 时，则当前块的程序执行将在该点终止，并且不执行 RET 指令以后的指令。 如果当前块为 OB，则参数“Return_Value”将被忽略。 如果当前块为 FC 或 FB，则将参数“Return_Value”的值作为被调用功能框的 ENO 值传回到调用例程。

不要求用户将 RET 指令用作块中的最后一个指令；该操作是自动完成的。 一个块中可以有多个 RET 指令。

以下是在 FC 代码块中使用 RET 指令的示例步骤：

1. 创建新项目并添加 FC：
2. 编辑该 FC：
   - 从指令树添加指令。
   - 添加一个 RET 指令，包括参数“Return_Value”的以下值之一：

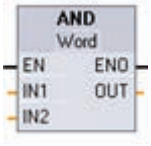
     TRUE、FALSE，或用于指定所需返回值的存储位置。
   - 添加更多的指令。
3. 从 MAIN [OB1] 调用 FC。

MAIN 代码块中 FC 功能框的 EN 输入必须为真，才能开始执行 FC。

执行了有能流通过 RET 指令的 FC 后，该 FC 的 RET 指令所指定的值将出现在 MAIN 代码块中 FC 功能框的 ENO 输出上。

6.1.9 逻辑运算

AND、OR 和 XOR 指令



AND： BYTE、WORD 和 DWORD 数据类型的逻辑与运算
OR： BYTE、WORD 和 DWORD 数据类型的逻辑或运算
XOR： BYTE、WORD 和 DWORD 数据类型的逻辑异或运算
在功能框名称下方单击，并从下拉菜单中选择数据类型。

参数	数据类型	说明
IN1、IN2	Byte、Word、DWord	逻辑输入
OUT	Byte、Word、DWord	逻辑输出

所选数据类型将 IN1、IN2 和 OUT 设置为相同的数据类型。 IN1 和 IN2 的相应位值相互组合，在参数 OUT 中生成二进制逻辑结果。 执行这些指令之后，ENO 总是为 TRUE。

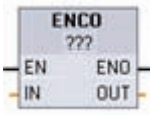
取反指令

INV 指令用于获得参数 IN 的二进制反码。通过对参数 IN 各位的值取反来计算反码（将每个 0 变为 1，每个 1 变为 0）。 执行该指令后，ENO 总是为 TRUE。

在功能框名称下方单击，并从下拉菜单中选择数据类型。

参数	数据类型	说明
IN	SInt、Int、DInt、USInt、UInt、UDInt、Byte、Word、DWord	要取反的数据元素
OUT	SInt、Int、DInt、USInt、UInt、UDInt、Byte、Word、DWord	取反后的输出

编码和解码指令



ENCO 将位序列编码成二进制数。

DECO 将二进制数解码成位序列。

在功能框名称下方单击，并从下拉菜单中选择数据类型。

参数	数据类型	说明
IN	ENCO： Byte、Word、DWord DECO：UInt	ENCO： 要编码的位序列 DECO： 要解码的值
OUT	ENCO：Int DECO： Byte、Word、DWord	ENCO： 编码后的值 DECO：解码后的位序列

ENCO 指令将参数 IN 转换为与参数 IN 的最低有效设置位的位位置对应的二进制数，并将结果返回给参数 OUT。 如果参数 IN 为 0000 0001 或 0000 0000，则将值 0 返回给 OUT。 如果参数 IN 的值为 0000 0000，则 ENO 被设置为 FALSE。

DECO 指令通过将参数 OUT 中的相应位位置设置为 1（其它所有位设置为 0），从参数 IN 解码二进制数。 执行 DECO 指令之后，ENO 始终为 TRUE。

DECO 参数 OUT 的数据类型选择（Byte、Word 或 DWord）会限制参数 IN 的可用范围。如果参数 IN 的值超出可用范围，将执行求模运算，如下所示提取最低有效位。

DECO 参数 IN 的范围：

- 3 位（值 0-7）IN 用于设置字节 OUT 中 1 的位位置
- 4 位（值 0-15）IN 用于设置字 OUT 中 1 的位位置
- 5 位（值 0-31）IN 用于设置双字 OUT 中 1 的位位置

DECO IN 值		DECO OUT 值（解码单个位位置）
		Byte OUT（8 位）：
最小 IN	0	00000001
最大 IN	7	10000000
		Word OUT（16 位）：
最小 IN	0	0000000000000001
最大 IN	15	1000000000000000
		DWord OUT： （32 位）：
最小 IN	0	00000000000000000000000000000001
最大 IN	31	10000000000000000000000000000000

ENO 状态	条件	结果 (OUT)
1	无错误	有效位号
0	IN 为零	OUT 被设置为零

选择 (SEL) 和多路复用 (MUX) 指令

- SEL 根据参数 G 的值将两个输入值之一分配给参数 OUT。
- MUX 根据参数 K 的值将多个输入值之一分配给参数 OUT。 如果参数 K 的值超出有效范围，则将参数 ELSE 的值分配给参数 OUT。

在功能框名称下方单击，并从下拉菜单中选择数据类型。

SEL	数据类型	说明
G	Bool	选择器开关： • FALSE 表示使用 IN0 的值 • TRUE 表示使用 IN1 的值
IN0、IN1	SInt、Int、DInt、USInt、UInt、UDInt、Real,、Byte、Word、DWord、Time、Char	输入
OUT	SInt、Int、DInt、USInt、UInt、UDInt、Real,、Byte、Word、DWord、Time、Char	输出

MUX	数据类型	说明
K	UInt	选择器的值： • 0 表示使用 IN0 的值 • 1 表示使用 IN1 的值 • ...
IN0、IN1 等	SInt、Int、DInt、USInt、UInt、UDInt、Real,、Byte、Word、DWord、Time、Char	输入
ELSE	SInt、Int、DInt、USInt、UInt、UDInt、Real,、Byte、Word、DWord、Time、Char	输入替换值（可选）
OUT	SInt、Int、DInt、USInt、UInt、UDInt、Real,、Byte、Word、DWord、Time、Char	输出

输入变量和输出变量必须为相同的数据类型。

- SEL 指令始终在两个 IN 值之间进行选择。
- MUX 指令在最初放置到程序编辑器中时有两个 IN 参数，但可对其进行扩展，以添加更多的 IN 参数。

使用以下方法可以为 MUX 指令添加和删除输入参数：

- 要添加输入，请在其中一个现有 IN 参数的输入短线处单击右键，并选择“插入输入”(Insert input) 命令。
- 要删除输入，请在其中一个现有 IN 参数（多于两个原始输入时）的输入短线处单击右键，并选择“删除”(Delete) 命令。

条件代码： 执行 SEL 指令之后，ENO 始终为 TRUE。

ENO 状态 (MUX)	MUX 条件	MUX 结果 (OUT)
1	无错误	选定的 IN 值被分配给 OUT
0	K 大于或等于 IN 参数的数量	未提供 ELSE： OUT 不变 提供了 ELSE： ELSE 的值被分配给 OUT

6.1.10 移位和循环

移位指令

移位指令用于将参数 IN 的位序列移位。结果分配给参数 OUT。 参数 N 指定移位的位数：

- SHR： 右移位序列
- SHL： 左移位序列

在功能框名称下方单击，并从下拉列表中选择数据类型。

参数	数据类型	说明
IN	Byte, Word, DWord	要移位的位序列
N	UInt	要移位的位数
OUT	Byte, Word, DWord	移位操作后的位序列

- N=0 时，不进行移位，并将 IN 值分配给 OUT。
- 用 0 填充移位操作清空的位位置。
- 如果要移位的位数 (N) 超过目标值中的位数（Byte 为 8 位、Word 为 16 位、DWord 为 32 位），则所有原始位值将被移出并用 0 代替（将 0 分配给 OUT）。
- 对于移位操作，ENO 总是为 TRUE。

Word 大小数据的 SHL 示例： 自左移入 0			
IN	1110 0010 1010 1101	首次移位前的 OUT 值：	1110 0010 1010 1101
		首次左移后：	1100 0101 0101 1010
		第二次左移后：	1000 1010 1011 0100
		第三次左移后：	0001 0101 0110 1000

循环指令

循环指令用于将参数 IN 的位序列循环移位。结果分配给参数 OUT。 参数 N 定义循环移位的位数。

- ROR：循环右移位序列
- ROL：循环左移位序列

在功能框名称下方单击，并从下拉菜单中选择数据类型。

参数	数据类型	说明
IN	Byte, Word, DWord	要循环移位的位序列
N	UInt	要循环移位的位数
OUT	Byte, Word, DWord	循环移位操作后的位序列

- N=0 时，不进行循环移位，并将 IN 值分配给 OUT。
- 从目标值一侧循环移出的位数据将循环移位到目标值的另一侧，因此原始位值不会丢失。
- 如果要循环移位的位数 (N) 超过目标值中的位数（Byte 为 8 位、Word 为 16 位、DWord 为 32 位），仍将执行循环移位。
- 执行循环指令之后，ENO 始终为 TRUE。

WORD 大小数据的 ROR 实例： 将各个位从右侧循环移出到左侧			
IN	0100 0000 0000 0001	首次循环移位前的 OUT 值：	0100 0000 0000 0001
		首次循环右移后：	1010 0000 0000 0000
		第二次循环右移后：	0101 0000 0000 0000

6.2 扩展指令

6.2.1 用于扩展指令的常见错误参数

扩展指令说明中介绍了各程序指令可能发生的运行错误。 除了这些错误，还可能发生下列常见错误。 如果执行代码块时发生某个常见错误，则 CPU 将进入 STOP 模式，除非在该代码块中使用 GetError 或 GetErrorID 指令编写程序来响应错误。

条件代码值 (W#16#....)	说明
8022	存储区对于输入太小
8023	存储区对于输出太小
8024	输入区非法
8025	输出区非法
8028	输入位赋值非法
8029	输出位赋值非法
8030	输出区是只读 DB
803A	DB 不存在

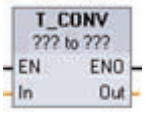
6.2.2 时钟和日历指令

日期和时间指令

日期和时间指令用于设计日历和时间计算。

- T_CONV 用于转换时间值的数据类型： （Time 转换为 DInt）或（DInt 转换为 Time）
- T_ADD 用于将 Time 与 DTL 值相加： (Time + Time = Time) 或 (DTL + Time = DTL)
- T_SUB 用于将 Time 与 DTL 值相减： (Time - Time = Time) 或 (DTL - Time = DTL)
- T_DIFF 提供两个 DTL 值的差作为 Time 值： DTL - DTL = Time

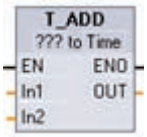
数据类型	大小（位）	有效范围
Time	32 存储形式	T#-24d_20h_31m_23s_648ms 到 T#24d_20h_31m_23s_647ms -2,147,483,648 ms 到 +2,147,483,647 ms
DTL 数据结构		
年：UInt	16	1970 到 2554
月：USInt	8	1 到 12
日：USInt	8	1 到 31
工作日：USInt	8	1 = 周日到 7 = 周六
小时：USInt	8	0 到 23
分钟：USInt	8	0 到 59
秒：USInt	8	0 到 59
纳秒：UDInt	32	0 到 999,999,999



T_CONV（时间转换）将 Time 数据类型转换为 DInt 数据类型，或将 DInt 数据类型转回 Time 数据类型。

参数	参数类型	数据类型	说明
IN	IN	DInt、Time	输入的 Time 值或 DInt 值
OUT	OUT	DInt、Time	转换后的 DInt 值或 Time 值

从指令名称下方提供的下拉列表中选择 IN 和 OUT 的数据类型。



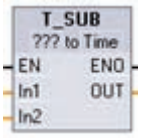
T_ADD（时间相加）将输入 IN1 的值（DTL 或 Time 数据类型）与输入 IN2 的 Time 值相加。 参数 OUT 提供 DTL 或 Time 值结果。

允许以下两种数据类型的运算：

- Time + Time = Time
- DTL + Time = DTL

参数	参数类型	数据类型	说明
IN1	IN	DTL、Time	DTL 或 Time 值
IN2	IN	Time	要加上的 Time 值
OUT	OUT	DTL、Time	DTL 或 Time 和值

从指令名称下方提供的下拉列表中选择 IN1 的数据类型。 所选的 IN1 数据类型同时也会设置参数 OUT 的数据类型。



T_SUB（时间相减）从 IN1（DTL 或 Time 值）中减去 IN2 的 Time 值。参数 OUT 以 DTL 或 Time 数据类型提供差值。

允许以下两种数据类型的运算：

- Time - Time = Time
- DTL - Time = DTL

参数	参数类型	数据类型	说明
IN1	IN	DTL、Time	DTL 或 Time 值
IN2	IN	Time	要减去的 Time 值
OUT	OUT	DTL、Time	DTL 或 Time 差值

从指令名称下方提供的下拉列表中选择 IN1 的数据类型。所选的 IN1 数据类型同时也会设置参数 OUT 的数据类型。

T_DIFF（时间差）从 IN1 DTL 值中减去 IN2 的 DTL 值。 参数 OUT 以 Time 数据类型提供差值。

- DTL - DTL = Time

参数	参数类型	数据类型	说明
IN1	IN	DTL	DTL 值
IN2	IN	DTL	要减去的 DTL 值
OUT	OUT	Time	Time 差值

条件代码： ENO = 1 表示未发生错误。 ENO = 0 和参数 OUT = 0 表示出现以下错误：

- DTL 值无效
- Time 值无效

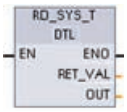
时钟指令

时钟指令用于设置和读取 PLC 系统时钟。 使用数据类型 DTL 提供日期和时间值。

DTL 结构	大小	有效范围
年：UInt	16 位	1970 到 2554
月：USInt	8 位	1 到 12
日：USInt	8 位	1 到 31
工作日：USInt	8 位	1 = 周日到 7 = 周六
小时：USInt	8 位	0 到 23
分钟：USInt	8 位	0 到 59
秒：USInt	8 位	0 到 59
纳秒：UDInt	32 位	0 到 999,999,999

WR_SYS_T（写入系统时间）使用参数 IN 中的 DTL 值设置 PLC 日时钟。该时间值不包括本地时区或夏令时偏移量。

参数	参数类型	数据类型	说明
IN	IN	DTL	要在 PLC 系统时钟内设置的日时钟
RET_VAL	OUT	Int	执行条件代码



RD_SYS_T（读取系统时间）从 PLC 读取当前系统时间。该时间值不包括本地时区或夏令时偏移量。

参数	参数类型	数据类型	说明
RET_VAL	OUT	Int	执行条件代码
OUT	OUT	DTL	当前 PLC 系统时间

RD_LOC_T（读取本地时间）以 DTL 数据类型提供 PLC 的当前本地时间。

参数	参数类型	数据类型	说明
RET_VAL	OUT	Int	执行条件代码
OUT	OUT	DTL	当地时间

- 通过使用用户在 CPU 时钟设备配置中设置的时区和夏令时偏移量计算本地时间。
- 时区组态是相对于协调世界时 (UTC, Coordinated Universal Time) 系统时间的偏移量。
- 夏令时组态指定夏令时开始时的月份、星期、日期和小时。
- 标准时间组态也会指定标准时间开始时的月份、星期、日期和小时。
- 时区偏移量始终会应用到系统时间值。 只有在夏令时有效时才会应用夏令时偏移量。

条件代码： ENO = 1 表示未发生错误。 ENO = 0 表示发生了执行错误，同时在 RET_VAL 输出中提供条件代码。

RET_VAL (W#16#....)	说明
0000	无错误
8080	本地时间不可用
8081	非法年份值

RET_VAL (W#16#....)	说明
8082	非法月份值
8083	非法日期值
8084	非法小时值
8085	非法分钟值
8086	非法秒数值
8087	非法纳秒值
80B0	实时时钟发生了故障

6.2.3 字符串和字符指令

6.2.3.1 String 数据概述

字符串数据类型

String 数据被存储成 2 个字节的标头后跟最多 254 个 ASCII 码字符组成的字符字节。 String 标头包含两个长度。 第一个字节是初始化字符串时方括号中给出的最大长度，默认值为 254。 第二个标头字节是当前长度，即字符串中的有效字符数。 当前长度必须小于或等于最大长度。 String 格式占用的存储字节数比最大长度大 2 个字节。

初始化 String 数据

在执行任何字符串指令之前，必须将 String 输入和输出数据初始化为存储器中的有效字符串。

有效 String 数据

有效字符串的最大长度必须大于 0 且小于 255。当前长度必须小于等于最大长度。

字符串无法分配给 I 或 Q 存储区。

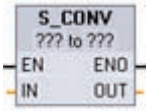
有关详细信息，请参见： String 数据类型的格式 (页 63)

6.2.3.2 字符串转换指令

字符串到值以及值到字符串的转换

可以使用以下指令将数字字符串转换为数值或将数值转换为数字字符串：

- S_CONV 用于将数字字符串转换成数值或将数值转换成数字字符串
- STRG_VAL 使用格式选项将数字字符串转换成数值
- VAL_STRG 使用格式选项将数值转换成数字字符串



S_CONV（字符串转换）将字符串转换成相应的值，或将值转换成相应的字符串。 S_CONV 指令没有输出格式选项。 因此，S_CONV 指令比 STRG_VAL 指令和 VAL_STRG 指令更简单，但灵活性更差。

从下拉列表中选择参数的数据类型。

S_CONV（字符串到值的转换）

参数	参数类型	数据类型	说明
IN	IN	String	输入字符串
OUT	OUT	String, SInt, Int, DInt, USInt, UInt, UDInt, Real	输出数值

字符串参数 IN 的转换从首个字符开始，并一直进行到字符串的结尾，或者一直进行到遇到第一个不是“0”到“9”、“+”、“-”或“.”的字符为止。 结果值将在参数 OUT 中指定的位置提供。 如果输出数值不在 OUT 数据类型的范围内，则参数 OUT 设置为 0，并且 ENO 设置为 FALSE。 否则，参数 OUT 将包含有效的结果，并且 ENO 设置为 TRUE。

输入 String 格式规则：

- 如果在 IN 字符串中使用小数点，则必须使用“.”字符。
- 允许使用逗点字符“,”作为小数点左侧的千位分隔符，并且逗点字符会被忽略。
- 忽略前导空格。
- 仅支持定点表示法。 字符“e”和“E”不会被识别为指数表示法。

S_CONV（值到字符串的转换）

参数	参数类型	数据类型	说明
IN	IN	String, SInt, Int, DInt, USInt, UInt, UDInt, Real	输入数值
OUT	OUT	String	输出字符串

整数值、无符号整数值或浮点值 IN 在 OUT 中被转换为相应的字符串。 在执行转换前，参数 OUT 必须引用有效字符串。 有效字符串由第一个字节中的最大字符串长度、第二个字节中的当前字符串长度以及后面字节中的当前字符串字符组成。 转换后的字符串将从第一个字符开始替换 OUT 字符串中的字符，并调整 OUT 字符串的当前长度字节。 OUT 字符串的最大长度字节不变。

被替换的字符数取决于参数 IN 的数据类型和数值。 被替换的字符数必须在参数 OUT 的字符串长度范围内。 OUT 字符串的最大字符串长度（第一个字节）应大于或等于被转换字符的最大预期数目。

下表列出了所支持的各种数据类型要求的最大可能字符串长度。

IN 数据类型	OUT 字符串中被转换字符的最大数目	实例	包括最大及当前长度字节在内的总字符串长度
USInt	3	255	5
SInt	4	-128	6
UInt	5	65535	7
Int	6	-32768	8
UDInt	10	4294967295	12
DInt	11	-2147483648	13

输出 String 格式规则：

- 写入到参数 OUT 的值不使用前导“+”号。
- 使用定点表示法（不可使用指数表示法）。
- 参数 IN 为 Real 数据类型时，使用句点字符“.”表示小数点。

STRG_VAL 指令

STRG_VAL（字符串到值）将数字字符串转换为相应的整型或浮点型表示法。 转换从字符串 IN 中的字符偏移量 P 位置开始，并一直进行到字符串的结尾，或者一直进行到遇到第一个不是“+”、“-”、“.”、“,”、“e”、“E”或“0”到“9”的字符为止。 结果放置在参数 OUT 中指定的位置。

同时，还将返回参数 P 作为原始字符串中转换终止位置的偏移量计数。 必须在执行前将 String 数据初始化为存储器中的有效字符串。

参数	参数类型	数据类型	说明
IN	IN	String	要转换的 ASCII 字符串
FORMAT	IN	Word	输出格式选项
P	IN_OUT	UInt	IN： 指向要转换的第一个字符的索引（第一个字符 = 1） OUT： 转换过程结束后，指向下一个字符的索引
OUT	OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real	转换后的数值

STRG_VAL 的 FORMAT 参数

以下定义了 STRG_VAL 指令的 FORMAT 参数。 未使用的位位置必须设置为零。

位 16							位 8	位 7							位 0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	f	r

f = 表示法格式　　1= 指数表示法
0 = 定点表示法

r = 小数点格式　　1 =“,”（逗点字符）
0 =“.”（句点字符）

FORMAT (W#16#)	表示法格式	小数点表示法
0000（默认）	定点	"."
0001		","
0002	指数	"."
0003		","
0004 到 FFFF	非法值	

STRG_VAL 转换的规则如下：

- 如果使用句点字符“.”作为小数点，则小数点左侧的逗点“,”将被解释为千位分隔符字符。 允许使用逗点字符并且会将其忽略。
- 如果使用逗点字符“,”作为小数点，则小数点左侧的句点“.”将被解释为千位分隔符字符。 允许使用句点字符并且会将其忽略。
- 忽略前导空格。

VAL_STRG 指令

VAL_STRG（值到字符串）将整数值、无符号整数值或浮点值转换为相应的字符串表示法。 参数 IN 表示的值将被转换为参数 OUT 所引用的字符串。 在执行转换前，参数 OUT 必须为有效字符串。

转换后的字符串将从字符偏移量计数 P 位置开始替换 OUT 字符串中的字符，一直到参数 SIZE 指定的字符数。 SIZE 中的字符数必须在 OUT 字符串长度范围内（从字符位置 P 开始计数）。该指令对于将数字字符嵌入到文本字符串中很有用。 例如，可以将数字“120”放入字符串“Pump pressure = 120 psi”中。

参数	参数类型	数据类型	说明
IN	IN	SInt、Int、DInt、USInt、UInt、UDInt、Real	要转换的值
SIZE	IN	USInt	要写入 OUT 字符串的字符数

参数	参数类型	数据类型	说明
PREC	IN	USInt	小数部分的精度或大小。 不包括小数点。
FORMAT	IN	Word	输出格式选项
P	IN_OUT	UInt	IN： 指向要替换的第一个 OUT 字符串字符的索引（第一个字符 = 1） OUT： 指向替换后的下一个 OUT 字符串字符的索引
OUT	OUT	String	转换后的字符串

参数 PREC 用于指定字符串中小数部分的精度或位数。 如果参数 IN 的值为整数，则 PREC 指定小数点的位置。 例如，如果数据值为 123 而 PREC = 1，则结果为“12.3”。 对于 REAL 数据类型支持的最大精度为 7 位。

如果参数 P 大于 OUT 字符串的当前大小，则会添加空格，一直到位置 P，并将该结果附加到字符串末尾。 如果达到了最大 OUT 字符串长度，则转换结束。

VAL_STRG 的 FORMAT 参数

以下定义了 VAL_STRG 指令的 FORMAT 参数。未使用的位位置必须设置为零。

位 16							位 8	位 7							位 0
0	0	0	0	0	0	0	0	0	0	0	0	0	s	f	r

s = 数字符号字符
1= 使用符号字符“+”和“-”
0 = 仅使用符号字符“-”

f = 表示法格式
1= 指数表示法
0 = 定点表示法

r = 小数点格式
1 =“,”（逗点字符）
0 =“.”（句点字符）

FORMAT (WORD)	数字符号字符	表示法格式	小数点表示法
W#16#0000	仅“-”	定点	"."
W#16#0001			","
W#16#0002		指数	"."
W#16#0003			","
W#16#0004	“+”和“-”	定点	"."
W#16#0005			","
W#16#0006		指数	"."
W#16#0007			","
W#16#0008 到 W#16#FFFF	非法值		

参数 OUT 字符串的格式规则如下：

- 如果转换后的字符串小于指定的大小，则会在字符串的最左侧添加前导空格字符。
- 如果 FORMAT 参数的符号位为 FALSE，则会将无符号和有符号整型值写入输出缓冲区，且不带前导“+”号。 必要时会使用“-”号。
  <前导空格><无前导零的数字>'.'<PREC 数字>
- 如果符号位为 TRUE，则会将无符号和有符号整型值写入输出缓冲区，且始终带有前导符号字符。
  <前导空格><符号><无前导零的数字>'.'<PREC 数字>
- 如果 FORMAT 被设置为指数表示法，则会按以下方式将 REAL 数据类型的值写入输出缓冲区：
  <前导空格><符号><数字> '.' <PREC 数字>'E' <符号><无前导零的数字>
- 如果 FORMAT 被设置为定点表示法，则会按以下方式将整型、无符号整型和实型值写入输出缓冲区：
  <前导空格><符号><无前导零的数字>'.'<PREC 数字>
- 小数点左侧的前导零会被隐藏，但与小数点相邻的数字除外。
- 小数点右侧的值被舍入为 PREC 参数所指定的小数点右侧的位数。
- 输出字符串的大小必须比小数点右侧的位数多至少三个字节。
- 输出字符串中的值为右对齐。

ENO 报告的条件

在转换操作期间遇到错误时，将返回以下结果：

- ENO 设置为 0。
- OUT 设置为 0，或者如字符串到值的转换实例中所示。
- OUT 不变，或者如 OUT 为字符串时的实例中所示。

ENO 状态	说明
1	无错误
0	非法或无效参数；例如，访问一个不存在的 DB
0	非法字符串，要求该字符串的最大长度为 0 或 255
0	非法字符串，当前长度大于最大长度
0	转换后的数值对于指定的 OUT 数据类型而言过大
0	OUT 参数的最大字符串大小必须足够大，以接受参数 SIZE 所指定的字符数（从字符位置参数 P 开始）
0	非法 P 值，P=0 或 P 大于当前字符串长度
0	参数 SIZE 必须大于参数 PREC

S_CONV 字符串到值的转换实例

IN 字符串	OUT 数据类型	OUT 值	ENO
"123"	Int/DInt	123	TRUE
"-00456"	Int/DInt	-456	TRUE
"123.45"	Int/DInt	123	TRUE
"+2345"	Int/DInt	2345	TRUE
"00123AB"	Int/DInt	123	TRUE
"123"	Real	123.0	TRUE
"123.45"	Real	123.45	TRUE
"1.23e-4"	Real	1.23	TRUE
"1.23E-4"	Real	1.23	TRUE
"12,345.67"	Real	12345.67	TRUE

IN 字符串	OUT 数据类型	OUT 值	ENO
"3.4e39"	Real	3.4	TRUE
"-3.4e39"	Real	-3.4	TRUE
"1.17549e-38"	Real	1.17549	TRUE
"12345"	SInt	0	FALSE
"A123"	不适用	0	FALSE
""	不适用	0	FALSE
"++123"	不适用	0	FALSE
"+-123"	不适用	0	FALSE

S_CONV 值到字符串的转换实例

数据类型	IN 值	OUT 字符串	ENO
UInt	123	"123"	TRUE
UInt	0	"0"	TRUE
UDInt	12345678	"12345678"	TRUE
Real	-INF	"INF"	FALSE
Real	+INF	"INF"	FALSE
Real	NaN	"NaN"	FALSE

STRG_VAL 转换实例

IN 字符串	FORMAT (W#16#....)	OUT 数据类型	OUT 值	ENO
"123"	0000	Int/DInt	123	TRUE
"-00456"	0000	Int/DInt	-456	TRUE
"123.45"	0000	Int/DInt	123	TRUE
"+2345"	0000	Int/DInt	2345	TRUE
"00123AB"	0000	Int/DInt	123	TRUE
"123"	0000	Real	123.0	TRUE

IN 字符串	FORMAT (W#16#....)	OUT 数据类型	OUT 值	ENO
"-00456"	0001	Real	-456.0	TRUE
"+00456"	0001	Real	456.0	TRUE
"123.45"	0000	Real	123.45	TRUE
"123.45"	0001	Real	12345.0	TRUE
"123,45"	0000	Real	12345.0	TRUE
"123,45"	0001	Real	123.45	TRUE
".00123AB"	0001	Real	123.0	TRUE
"1.23e-4"	0000	Real	1.23	TRUE
"1.23E-4"	0000	Real	1.23	TRUE
"1.23E-4"	0002	Real	1.23E-4	TRUE
"12,345.67"	0000	Real	12345.67	TRUE
"12,345.67"	0001	Real	12.345	TRUE
"3.4e39"	0002	Real	+INF	TRUE
"-3.4e39"	0002	Real	-INF	TRUE
"1.1754943e-38"（及更小值）	0002	Real	0.0	TRUE
"12345"	不适用	SInt	0	FALSE
"A123"	不适用	不适用	0	FALSE
""	不适用	不适用	0	FALSE
"++123"	不适用	不适用	0	FALSE
"+-123"	不适用	不适用	0	FALSE

VAL_STRG 转换实例

下面的实例均基于按以下方式初始化的 OUT 字符串：

“Current Temp = xxxxxxxxxx C”
字符“x”表示为已转换值分配的空格字符。

数据类型	IN 值	P	SIZE	FORMAT (W#16#....)	PREC	OUT 字符串	ENO
UInt	123	16	10	0000	0	Current Temp = xxxxxxx123 C	TRUE
UInt	0	16	10	0000	2	Current Temp = xxxxxx0.00 C	TRUE
UDInt	12345678	16	10	0000	3	Current Temp = x12345.678 C	TRUE
UDInt	12345678	16	10	0001	3	Current Temp = x12345,678 C	TRUE
Int	123	16	10	0004	0	Current Temp = xxxxxx+123 C	TRUE
Int	-123	16	10	0004	0	Current Temp = xxxxxx-123 C	TRUE
Real	-0.00123	16	10	0004	4	Current Temp = xxx-0.0012 C	TRUE
Real	-0.00123	16	10	0006	4	Current Temp = -1.2300E-3 C	TRUE
Real	-INF	16	10	不适用	4	Current Temp = xxxxxx-INF C	FALSE
Real	+INF	16	10	不适用	4	Current Temp = xxxxxx+INF C	FALSE
Real	NaN	16	10	不适用	4	Current Temp = xxxxxxxNaN C	FALSE
UDInt	12345678	16	6	不适用	3	Current Temp = xxxxxxxxxx C	FALSE

6.2.3.3 字符串操作指令

控制程序可以使用以下字符串和字符指令为操作员显示和过程日志创建消息。

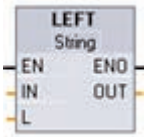
所有 String 操作的常见错误

存在下述非法或无效 String 条件时执行 String 操作指令将导致 ENO = 0 和字符串输出为空。 针对特定指令出现的错误条件列在相应的指令操作说明下面。

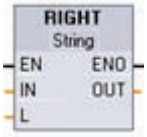
ENO	条件	OUT
0	IN1 的当前长度超出 IN1 的最大长度，或者 IN2 的当前长度超出 IN2 的最大长度（无效字符串）	当前长度被设置为 0
	IN1、IN2 或 OUT 的最大长度不在分配的存储范围内	
	IN1、IN2 或 OUT 的最大长度为 0 或 255（非法长度）	

LEN：获取字符串长度

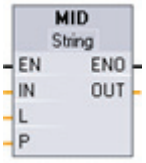
CONCAT：连接两个字符串



LEFT：获取字符串的左侧子串

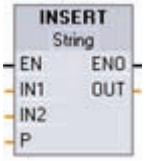


RIGHT：获取字符串的右侧子串



MID：获取字符串的中间子串

FIND：查找字符串中的子串或字符



INSERT：在字符串中插入子串

DELETE：删除字符串的子串

REPLACE：替换字符串中的子串

LEN 指令

参数	参数类型	数据类型	说明
IN	IN	String	输入字符串
OUT	OUT	UInt	IN 字符串的有效字符数

LEN（字符串长度）在输出 OUT 端给出字符串 IN 的当前长度。 空字符串的长度为零。下表列出了指令的条件代码。

ENO	条件	OUT
1	没有无效字符串条件	有效字符串长度

CONCAT 指令

参数	参数类型	数据类型	说明
IN1	IN	String	输入字符串 1
IN2	IN	String	输入字符串 2
OUT	OUT	String	组合字符串（字符串 1 + 字符串 2）

CONCAT（连接字符串）连接 String 参数 IN1 和 IN2 以形成一个在 OUT 端提供的字符串。 连接后，字符串 IN1 是组合字符串的左侧部分而 IN2 是其右侧部分。 下表列出了指令的条件代码。

ENO	条件	OUT
1	未检测到错误	有效字符
0	连接后的结果字符串比 OUT 字符串的最大长度长	复制结果字符串字符直到达到 OUT 的最大长度为止

LEFT 指令

参数	参数类型	数据类型	说明
IN	IN	String	输入字符串
L	IN	Int	要使用 IN 字符串最左侧的 L 个字符创建的子串的长度
OUT	OUT	String	输出字符串

LEFT（左侧子串）提供字符串参数 IN 的前 L 个字符组成的子串。

- 如果 L 大于 IN 字符串的当前长度，则在 OUT 中返回整个 IN 字符串。
- 如果输入是空字符串，则在 OUT 中返回空字符串。

下表列出了指令的条件代码。

ENO	条件	OUT
1	未检测到错误	有效字符
0	L 小于或等于 0	当前长度被设置为 0
	要复制的子串长度 (L) 比 OUT 字符串的最大长度长	复制字符直到达到 OUT 的最大长度为止

RIGHT 指令

参数	参数类型	数据类型	说明
IN	IN	String	输入字符串
L	IN	Int	要使用 IN 字符串最右侧的 L 个字符创建的子串的长度
OUT	OUT	String	输出字符串

RIGHT（右侧子串）提供字符串的最后 L 个字符。

- 如果 L 大于 IN 字符串的当前长度，则在参数 OUT 中返回整个 IN 字符串。
- 如果输入是空字符串，则在 OUT 中返回空字符串。

下表列出了指令的条件代码。

ENO	条件	OUT
1	未检测到错误	有效字符
0	L 小于或等于 0	当前长度被设置为 0
	要复制的子串长度 (L) 比 OUT 字符串的最大长度长	复制字符直到达到 OUT 的最大长度为止

MID 指令

参数	参数类型	数据类型	说明
IN	IN	String	输入字符串
L	IN	Int	要使用 IN 字符串中从字符位置 P 开始的 L 个字符创建的子串的长度
P	IN	Int	要复制的第一个子串字符的位置： P= 1，表示 IN 字符串的初始字符位置
OUT	OUT	String	输出字符串

MID（中间子串）提供字符串的中间部分。 中间子串为从字符位置 P（包括该位置）开始的 L 个字符的长度。

如果 L 和 P 的和超出 String 参数 IN 的当前长度，则返回从字符位置 P 开始并一直到 IN 字符串结尾的子串。 下表列出了指令的条件代码。

ENO	条件	OUT
1	未检测到错误	有效字符
0	L 或 P 小于或等于 0	当前长度被设置为 0
	P 大于 IN 的最大长度	
	要复制的子串长度 (L) 比 OUT 字符串的最大长度长	从位置 P 开始复制字符直到达到 OUT 的最大长度为止

DELETE 指令

参数	参数类型	数据类型	说明
IN	IN	String	输入字符串
L	IN	Int	要删除的字符数
P	IN	Int	要删除的第一个字符的位置： IN 字符串的第一个字符的位置编号为 1
OUT	OUT	String	输出字符串

DELETE（删除子串）从字符串 IN 删除 L 个字符。从字符位置 P（包括该位置）开始删除字符，并在参数 OUT 中提供剩余子串。

- 如果 L 等于零，则在 OUT 中返回输入字符串。
- 如果 L 和 P 的和大于输入字符串的长度，则一直删除到该字符串的末尾。

下表列出了指令的条件代码。

ENO	条件	OUT
1	未检测到错误	有效字符
0	P 大于 IN 的当前长度	将 IN 复制到 OUT 且不删除任何字符
	L 小于 0，或者 P 小于或等于 0	当前长度被设置为 0
	删除字符后的结果字符串比 OUT 字符串的最大长度长	复制结果字符串字符直到达到 OUT 的最大长度为止

INSERT

参数	参数类型	数据类型	说明
IN1	IN	String	输入字符串 1
IN2	IN	String	输入字符串 2
P	IN	Int	字符串 IN1 中字符串 IN2 插入点前的最后一个字符位置。 字符串 IN1 的第一个字符的位置编号为 1。
OUT	OUT	String	结果字符串

INSERT（插入子串）将字符串 IN2 插入字符串 IN1 中。 在位置 P 的字符后开始插入。下表列出了指令的条件代码。

ENO	条件	OUT
1	未检测到错误	有效字符
0	P 大于 IN1 的长度	IN2 紧接最后一个 IN1 字符与 IN1 连接
	P 小于或等于 0	当前长度被设置为 0
	插入后的结果字符串比 OUT 字符串的最大长度长	复制结果字符串字符直到达到 OUT 的最大长度为止

REPLACE

参数	参数类型	数据类型	说明
IN1	IN	String	输入字符串
IN2	IN	String	替换字符的字符串
L	IN	Int	要替换的字符数
P	IN	Int	要替换的第一个字符的位置
OUT	OUT	String	结果字符串

REPLACE（替换子串）替换字符串参数 IN1 中的 L 个字符。 使用字符串参数 IN2 中的替换字符，从字符串 IN1 的字符位置 P（包括该位置）开始替换。

- 如果参数 L 等于零，则在字符串 IN1 的位置 P 插入字符串 IN2 而不从字符串 IN1 删除任何字符。
- 如果 P 等于 1，则使用字符串 IN2 字符替换字符串 IN1 的前 L 个字符。

下表列出了指令的条件代码。

ENO	条件	OUT
1	未检测到错误	有效字符
0	P 大于 IN1 的长度	IN2 紧接最后一个 IN1 字符与 IN1 连接
	P 小于 IN1 的长度，但 IN1 中没有 L 个字符	IN2 从位置 P 开始替换 IN1 的后端字符
	L 小于 0，或者 P 小于或等于 0	当前长度被设置为 0
	替换后的结果字符串比 OUT 字符串的最大长度长	复制结果字符串字符直到达到 OUT 的最大长度为止

FIND

参数	参数类型	数据类型	说明
IN1	IN	String	在该字符串内搜索
IN2	IN	String	搜索该字符串
OUT	OUT	Int	字符串 IN1 中第一个搜索匹配项的字符位置

FIND（查找子串）提供通过 IN2 所指定子串或字符在字符串 IN1 中的字符位置。 从左侧开始搜索。 在 OUT 中返回 IN2 字符串第一次出现的字符位置。 如果在字符串 IN1 中没有找到字符串 IN2，则返回零。 下表列出了指令的条件代码。

ENO	条件	OUT
1	未检测到错误	有效字符位置
0	IN2 大于 IN1	字符位置被设置为 0

6.2.4 程序控制指令

6.2.4.1 复位扫描循环监视狗指令

RE_TRIGR（重新触发扫描时间监视狗）用于延长扫描循环监视狗定时器生成错误前允许的最大时间。

RE_TRIGR 指令用于在单个扫描循环期间重新启动扫描循环定时器。 结果是从最后一次执行 RE_TRIGR 功能开始，使允许的最大扫描周期延长一个最大循环时间段。

CPU 只允许将 RE_TRIGR 指令用于程序循环，例如，OB1 和从该程序循环调用的功能。 也就是说，如果从程序循环 OB 列表的任何 OB 调用 RE_TRIGR，都会复位监视狗定时器且 ENO = EN。

如果从启动 OB、中断 OB 或错误 OB 执行 RE_TRIGR，则不会复位监视狗定时器且 ENO = FALSE。

设置 PLC 最大循环时间

可以在 PLC 设备配置中为“循环时间”(Cycle time) 设置最大扫描周期。

循环时间监视	最小值	最大值	默认值
最大循环时间	1 ms	6000 ms	150 ms

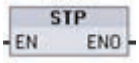
监视狗超时

如果最大扫描循环定时器在扫描循环完成前达到预置时间，则会生成错误。 如果用户程序中包含错误处理代码块 OB80，则 PLC 将执行 OB80，用户可以在其中添加程序逻辑以创建具体响应。 如果不包含 OB80，则忽略第一个超时条件。

如果在同一程序扫描中第二次发生最大扫描时间超时（2 倍的最大循环时间值），则触发错误导致 PLC 切换到 STOP 模式。

在 STOP 模式下，用户程序停止执行而 PLC 系统通信和系统诊断仍继续执行。

6.2.4.2 停止扫描循环指令



STP（停止 PLC 扫描循环）将 PLC 置于 Stop 模式。 PLC 处于 Stop 模式时，将停止程序执行以及停止过程映像的物理更新。

有关详细信息，请参见：组态从 RUN 切换到 STOP 时的输出 (页 52)

如果 EN = TRUE，PLC 将进入 STOP 模式，程序执行停止并且 ENO 状态无意义。 否则，EN = ENO = 0。

6.2.4.3 获取错误指令

获取错误指令提供有关程序块执行错误的信息。 如果在代码块中添加了 GetError 或 GetErrorID 指令，则可在程序块中处理程序错误。

GET_ERROR

GET_ERROR 指示发生程序块执行错误并用详细错误信息填充预定义的错误数据结构。

参数	数据类型	说明
ERROR	ErrorStruct	错误数据结构： 可以重命名该结构，但不能重命名结构中的成员。

ErrorStruct 数据元素	数据类型	说明
ERROR_ID	Word	错误标识符
FLAGS	Byte	始终设置为 0。
REACTION	Byte	对错误的响应： • 0 = 忽略；不执行写入（写入错误） • 1 = 替换： 0 用于输入值（读取错误） • 2 = 跳过该指令
BLOCK_TYPE	Byte	出错的块类型： • 1 = OB • 2 = FC • 3 = FB
PAD_0	Byte	用于调整的内部填充字节，将为 0
CODE_BLOCK_NUMBER	UInt	出错的块编号
ADDRESS	UDInt	出错指令的内部存储位置
MODE	Byte	如何解释剩余域以便 STEP 7 Basic 可以使用的内部映射

ErrorStruct 数据元素	数据类型	说明
PAD_1	Byte	用于调整的内部填充字节；如不使用，将为 0
OPERAND_NUMBER	UInt	内部指令操作数编号
POINTER_NUMBER_LOCATION	UInt	(A) 内部指令指针位置
SLOT_NUMBER_SCOPE	UInt	(B) 内部存储器存储位置
AREA	Byte	(C) 出错时引用的存储区： • L： 16#40 – 4E、86、87、8E、8F、C0 – CE • I： 16#81 • Q： 16#82 • M： 16#83 • DB： 16#84、85、8A、8B
PAD_2	Byte	用于调整的内部填充字节；如不使用，将为 0
DB_NUMBER	UInt	(D) 发生数据块错误时引用的数据块，否则为 0
OFFSET	UDInt	(E) 出错时引用的位偏移量（例如： 12 = 字节 1，位 4）

GET_ERR_ID

GET_ERR_ID 指示发生程序块执行错误并报告错误的 ID（标识符代码）。

参数	数据类型	说明
ID	Word	ErrorStruct ERROR_ID 成员的错误标识符值

ERROR_ID 十六进制值	ERROR_ID 十进制值	程序块执行错误
2503	9475	未初始化指针错误
2522	9506	操作数超出范围读取错误
2523	9507	操作数超出范围写入错误
2524	9508	无效区域读取错误
2525	9509	无效区域写入错误
2528	9512	数据分配读取错误（位赋值不正确）
2529	9513	数据分配写入错误（位赋值不正确）
2530	9520	DB 受到写保护
253A	9530	全局 DB 不存在
253C	9532	版本错误或 FC 不存在
253D	9533	指令不存在
253E	9534	版本错误或 FB 不存在
253F	9535	指令不存在
2575	9589	程序嵌套深度错误
2576	9590	局部数据分配错误
2942	10562	物理输入点不存在
2943	10563	物理输出点不存在

操作

默认情况下，CPU 通过将错误记录到诊断缓冲区并切换到 STOP 模式来响应块执行错误。 但是，如果在代码块中放置一个或多个 GET_ERROR 或 ERR_ID 指令，即将该块设置为在块内处理错误。 在这种情况下，CPU 不会切换到 STOP 模式且不会在诊断缓冲区中记录错误。 而是在 GET_ERROR 或 GET_ERR_ID 指令的输出中报告错误信息。 可以使用 GET_ERROR 指令读取详细错误信息，或使用 GET_ERR_ID 指令只读取错误标识符。 因为后续错误往往只是第一个错误的结果，所以第一个错误通常最重要。

在块内第一次执行 GET_ERROR 或 GET_ERR_ID 指令将返回块执行期间检测到的第一个错误。 在块启动到执行 GET_ERROR 或 GET_ERR_ID 期间随时都可能发生该错误。随后执行 GET_ERROR 或 GET_ERR_ID 将返回上次执行 GET_ERROR 或 GET_ERR_ID 以来发生的第一个错误。不保存错误历史，执行任一指令都将使 PLC 系统重新捕捉下一个错误。

可以在数据块编辑器和块接口编辑器中添加 GET_ERROR 指令所使用的 ErrorStruct 数据类型，从而程序逻辑可以访问这些值。 从数据类型下拉列表中选择 ErrorStruct 以添加该结构。 您可以使用唯一的名称创建多个 ErrorStruct。 不能重命名 ErrorStruct 的成员。

ENO 指示的错误条件

如果 EN = TRUE 且 GET_ERROR 或 GET_ERR_ID 执行，则：

- ENO = TRUE 表示发生代码块执行错误并提供错误数据
- ENO = FALSE 表示未发生代码块执行错误

可以将错误响应程序逻辑连接到在发生错误后激活的 ENO。 如果存在错误，该输出参数会将错误数据存储在程序能够访问这些数据的位置。

GET_ERROR 和 GET_ERR_ID 可用来将错误信息从当前执行块（被调用块）发送到调用块。 将该指令放置在被调用块程序的最后一个程序段中可以报告被调用块的最终执行状态。

6.2.5 通信指令

6.2.5.1 开放式以太网通信

可自动连接/断开的开放式以太网通信（TSEND_C 和 TRCV_C）

说明

处理 TSEND_C 和 TRCV_C 指令花费的时间量无法确定。 要确保这些指令在每次扫描循环中都被处理，务必从主程序循环扫描中对其调用，例如，从程序循环 OB 中或从程序循环扫描中调用的代码块中对其调用。 **不要**从硬件中断 OB、延时中断 OB、循环中断 OB、错误中断 OB 或启动 OB 调用这些指令。

有关使用这些指令传送数据的信息，请参阅数据一致性 (页 94)部分。

TSEND_C 描述

TSEND_C 可与伙伴站建立 TCP 或 ISO on TCP 通信连接，发送数据并且可以终止该连接。 设置并建立连接后，CPU 会自动保持和监视该连接。 TSEND_C 兼具 TCON、TDISCON 和 TSEND 的功能。

使用 TSEND_C 指令可以传送的最小数据单位是字节。

说明

LEN 参数的默认设置 (LEN = 0) 使用 DATA 参数来确定要传送的数据的长度。 确保 TSEND_C 指令传送的 DATA 的大小与 TRCV_C 指令的 DATA 参数的大小相同。

下列功能说明了 TSEND_C 指令的操作：

- 要建立连接，请在 CONT = 1 时执行 TSEND_C。
- 成功建立连接后，TSEND_C 便会置位 DONE 参数一个周期。
- 要终止通信连接，请在 CONT = 0 时执行 TSEND_C。连接将立即中止。 这还会影响接收站。 将在接收站关闭该连接，并且接收缓冲区内的数据可能会丢失。
- 要通过建立的连接发送数据，请在 REQ 的上升沿执行 TSEND_C。 发送操作成功执行后，TSEND_C 便会设置 DONE 参数一个周期。
- 要建立连接并发送数据，请在 CONT =1 且 REQ = 1 时执行 TSEND_C。发送操作成功执行后，TSEND_C 便会置位 DONE 参数一个周期。

TRCV_C 描述

TRCV_C 可与伙伴 CPU 建立 TCP 或 ISO on TCP 通信连接，接收数据并且可以终止该连接。设置并建立连接后，CPU 会自动保持和监视该连接。 TRCV_C 指令兼具 TCON、TDISCON 和 TRCV 指令的功能。

使用 TRCV_C 指令可以接收的最小数据单位是字节。 TRCV_C 指令不支持传送布尔数据或布尔数组。

说明

LEN 参数的默认设置 (LEN = 0) 使用 DATA 参数来确定要传送的数据的长度。 确保 TSEND_C 指令传送的 DATA 的大小与 TRCV_C 指令的 DATA 参数的大小相同。

下列功能说明了 TRCV_C 指令的操作：

- 要建立连接，请在参数 CONT = 1 时执行 TRCV_C。
- 要接收数据，请在参数 EN_R = 1 时执行 TRCV_C。参数 EN_R = 1 且 CONT = 1 时 TRCV_C 连续接收数据。
- 要终止连接，请在参数 CONT = 0 时执行 TRCV_C。连接将立即中止且数据可能丢失。

接收模式

TRCV_C 处理与 TRCV 指令相同的接收模式。 下表说明了在接收区输入数据的方法。

协议选项	在接收区输入数据	参数“connection_type”
TCP	指定长度的数据接收	B#16#11
ISO on TCP	协议控制	B#16#12

说明

由于 TSEND_C 采用异步处理，所以在 DONE 参数值或 ERROR 参数值为 TRUE 前，必须保持发送方区域中的数据一致。

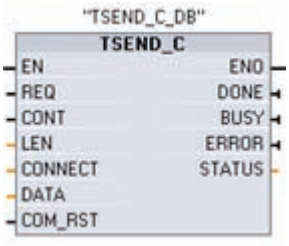
对于 TSEND_C，DONE 参数状态为 TRUE 表示数据成功发送。 但并不表示连接伙伴 CPU 实际读取了接收缓冲区。

由于 TRCV_C 采用异步处理，因此仅当参数 DONE = 1 时，接收器区域中的数据才一致。

下表说明了参数 BUSY、DONE 和 ERROR 之间的关系。

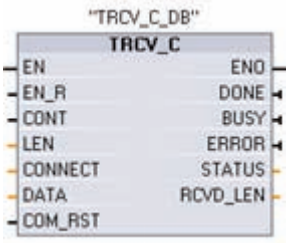
BUSY	DONE	ERROR	说明
TRUE	不相关	不相关	作业正在处理。
FALSE	TRUE	FALSE	作业已成功完成。
FALSE	FALSE	TRUE	该作业以出错而结束。 出错原因可在 STATUS 参数中找到。
FALSE	FALSE	FALSE	未分配新作业。

TSEND_C 参数



参数	参数类型	数据类型	说明
REQ	INPUT	Bool	控制参数 REQ 在上升沿启动具有 CONNECT 中所述连接的发送作业。
CONT	INPUT	Bool	• 0: 断开 • 1: 建立并保持连接
LEN	INPUT	Int	要发送的最大字节数。（默认值 = 0，这表示 DATA 参数决定要发送的数据的长度）。
CONNECT	IN_OUT	TCON-Param	指向连接描述的指针
DATA	IN_OUT	Variant	发送区；包含要发送数据的地址和长度。
COM_RST	IN_OUT	Bool	• 1: 完成功能块的重新启动，现有连接将终止。
DONE	OUTPUT	Bool	• 0: 作业尚未开始或仍在运行。 • 1: 无错执行作业。
BUSY	OUTPUT	Bool	• 0: 作业完成。 • 1: 作业尚未完成。 无法触发新作业。
ERROR	OUTPUT	Bool	• 1: 处理时出错。 STATUS 提供错误类型的详细信息。
STATUS	OUTPUT	Word	错误信息

TRCV_C 参数



参数	参数类型	数据类型	说明
EN_R	IN	Bool	启用接收的控制参数：EN_R = 1 时，TRCV_C 准备接收。 处理接收作业。
CONT	IN	Bool	控制参数 CONT： • 0: 断开 • 1: 建立并保持连接
LEN	IN	Int	接收区长度（字节）。 （默认值 = 0，这表示 DATA 参数决定要发送的数据的长度）。
CONNECT	IN_OUT	TCON-Param	指向连接描述的指针
DATA	IN_OUT	Variant	接收区包含接收数据的起始地址和最大长度。
COM_RST	IN_OUT	Bool	• 1: 完成功能块的重新启动，现有连接将终止。
DONE	OUT	Bool	• 0: 作业尚未开始或仍在运行。 • 1: 无错执行作业。
BUSY	OUT	Bool	• 0: 作业完成。 • 1: 作业尚未完成。 无法触发新作业。
ERROR	OUT	Bool	• 1: 处理时出错。 STATUS 提供错误类型的详细信息。
STATUS	OUT	Word	错误信息
RCVD_LEN	OUT	Int	实际接收到的数据量（字节）

参数 Error 和 Status

ERROR	STATUS (W#16#...)	说明
0	0000	作业已无错执行
0	7000	无激活的作业处理
0	7001	启动作业处理，正在建立连接，正在等待连接伙伴
0	7002	正在发送或接收数据
0	7003	正终止连接

ERROR	STATUS (W#16#...)	说明
0	7004	连接已建立并受到监视，无激活的作业处理
1	8085	LEN 参数的值比最大的允许值大
1	8086	CONNECT 参数超出允许范围
1	8087	已达到最大连接数；无法建立更多连接
1	8088	LEN 参数大于 DATA 中指定的存储区；接收存储区过小
1	8089	参数 CONNECT 未指向数据块。
1	8091	超出最大嵌套深度
1	809A	CONNECT 参数指向的域与连接描述的长度不匹配。
1	809B	连接描述中的 local_device_id 与 CPU 的不匹配。
1	80A1	通信错误： • 尚未建立指定的连接 • 当前正在终止指定的连接；无法通过该连接传输 • 正在重新初始化接口
1	80A3	正在尝试终止不存在的连接
1	80A4	远程伙伴连接的 IP 地址无效。 例如，远程伙伴的 IP 地址与本地伙伴的 IP 地址相同。
1	80A7	通信错误： 在 TCON 完成前调用了 TDISCON（TDISCON 必须先完全终止 ID 引用的连接）
1	80B2	参数 CONNECT 指向使用关键字 UNLINKED 生成的数据块
1	80B3	不一致的参数： • 连接描述错误 • 本地端口（参数 local_tsap_id）已在另一个连接描述中存在 • 连接描述中的 ID 与作为参数指定的 ID 不同

ERROR	STATUS (W#16#...)	说明
1	80B4	使用 ISO on TCP (connection_type = B#16#12) 建立被动连接时，条件代码 80B4 提示您输入的 TSAP 不符合下列某一项地址要求： • 若是本地 TSAP 长度为 2 个字节且首字节的 TSAP ID 值为 E0 或 E1（十六进制），第二字节必须为 00 或 01。 • 如果本地 TSAP 长度为 3 个或更多字节，且首字节的 TSAP ID 值为 E0 或 E1（十六进制），则第二字节必须为 00 或 01，且所有其它字节必须为有效的 ASCII 字符。 • 如果本地 TSAP 长度为 3 个或更多字节，且首字节的 TSAP ID 值既不为 E0 也不为 E1（十六进制），则 TSAP ID 的所有字节都必须为有效的 ASCII 字符。 有效 ASCII 字符的字节值为 20 到 7E（十六进制）。
1	80C3	所有连接资源都在使用。
1	80C4	临时通信错误： • 此时无法建立连接 • 接口正在接收新参数 • TDISCON 当前正在删除已组态连接
1	8722	CONNECT 参数： 源区域无效： DB 中不存在该区域
1	873A	CONNECT 参数： 无法访问连接描述（例如，DB 不可用）
1	877F	CONNECT 参数： 内部错误，如无效 ANY 引用

具有连接/断开控制的开放式以太网通信

说明

处理 TCON、TDISCON、TSEND 和 TRCV 指令花费的时间量无法确定。 要确保这些指令在每次扫描循环中都被处理，务必从主程序循环扫描中对其调用，例如，从程序循环 OB 中或从程序循环扫描中调用的代码块中对其调用。 **不要**从硬件中断 OB、延时中断 OB、循环中断 OB、错误中断 OB 或启动 OB 调用这些指令。

使用 TCP 和 ISO on TCP 协议的以太网通信

以下这些程序指令控制通信过程：

- TCON 建立连接。
- TSEND 和 TRCV 发送和接收数据。
- TDISCON 断开连接。

使用 TSEND 和 TRCV 指令可以传送或接收的最小数据单位是字节。 TRCV 指令不支持传送布尔数据或布尔数组。 有关使用这些指令传送数据的信息，请参阅数据一致性 (页 94)部分。

说明

LEN 参数的默认设置 (LEN = 0) 使用 DATA 参数来确定要传送的数据的长度。 确保 TSEND 指令传送的 DATA 的大小与 TRCV 指令的 DATA 参数的大小相同。

两个通信伙伴都执行 TCON 指令来设置和建立通信连接。 用户使用参数指定主动和被动通信端点伙伴。 设置并建立连接后，CPU 会自动保持和监视该连接。

例如，如果连接由于断线或远程通信伙伴而终止，主动伙伴会尝试重新建立组态的连接。不必再次执行 TCON。

执行 TDISCON 指令或 CPU 进入 STOP 模式后，会终止现有连接并删除所设置的连接。要设置和重新建立连接，必须再次执行 TCON。

功能说明

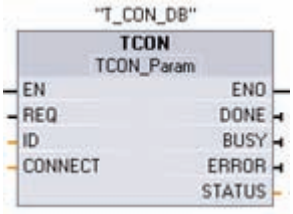
TCON、TDISCON、TSEND 和 TRCV 异步运行，即，作业处理需要使用多个指令执行来完成。

例如，在参数 REQ = 1 时执行指令 TCON 来启动用于设置和建立连接的作业。然后，再执行 TCON 来监视作业进度并使用参数 DONE 来测试作业是否已完成。

下表给出了 BUSY、DONE 和 ERROR 之间的关系。 使用该表可以确定当前作业状态。

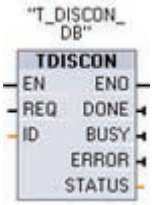
BUSY	DONE	ERROR	说明
TRUE	不相关	不相关	作业正在处理。
FALSE	TRUE	FALSE	作业已成功完成。
FALSE	FALSE	TRUE	该作业以出错而结束。出错原因可在 STATUS 参数中找到。
FALSE	FALSE	FALSE	未分配新作业。

TCON



参数	参数类型	数据类型	说明
REQ	IN	Bool	控制参数 REQUEST 启动用于建立连接的作业，该连接是通过 ID 指定的。在上升沿启动该作业。
ID	IN	CONN_OUC (Word)	引用要建立的、连接到远程伙伴或在用户程序和操作系统通信层之间的连接。 标识号必须与本地连接描述中的相关参数标识号相同。 值范围： W#16#0001 到 W#16#0FFF
CONNECT	IN_OUT	TCON-Param	指向连接描述的指针
DONE	OUT	Bool	状态参数 DONE： • 0: 作业尚未启动或仍在运行 • 1: 作业已无错执行
BUSY	OUT	Bool	BUSY = 1： 作业尚未完成 BUSY = 0： 作业已完成
ERROR	OUT	Bool	状态参数 ERROR： ERROR = 1： 作业处理期间出错。 STATUS 提供错误类型的详细信息。
STATUS	OUT	Word	状态参数 STATUS：错误信息

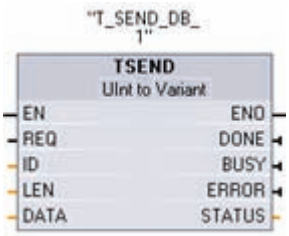
TDISCON



TCP 和 ISO on TCP： TDISCON 终止从 CPU 到通信伙伴的通信连接。

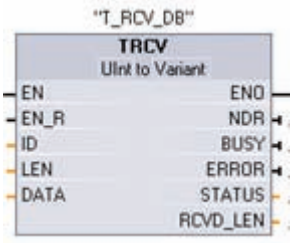
参数	参数类型	数据类型	说明
REQ	IN	Bool	控制参数 REQUEST 启动用于建立连接的作业，该连接是通过 ID 指定的。在上升沿启动该作业。
ID	IN	CONN_OUC (Word)	引用要终止的、连接到远程伙伴或在用户程序和操作系统通信层之间的连接。标识号必须与本地连接描述中的相关参数标识号相同。 值范围： W#16#0001 到 W#16#0FFF
DONE	OUT	Bool	状态参数 DONE： • 0: 作业尚未启动或仍在运行 • 1: 作业已无错执行
BUSY	OUT	Bool	BUSY = 1：作业尚未完成 BUSY = 0：作业已完成
ERROR	OUT	Bool	ERROR = 1：处理时出错。
STATUS	OUT	Word	错误代码

TSEND



参数	参数类型	数据类型	说明
REQ	IN	Bool	控制参数 REQUEST 在上升沿启动发送作业。 传送通过 LEN 和 DATA 指定的区域中的数据。
ID	IN	CONN_OUC (Word)	引用相关的连接。标识号必须与本地连接描述中的相关参数标识号相同。 值范围： W#16#0001 到 W#16#0FFF
LEN	IN	Int	要通过作业发送的最大字节数
DATA	IN_OUT	Variant	指向要发送数据区的指针： 发送方区域；包含地址和长度。 地址将参考： • 过程映像输入表 • 过程映像输出表 • 位存储器 • 数据块
DONE	OUT	Bool	状态参数 DONE： • 0: 作业尚未开始或仍在运行。 • 1: 无错执行作业。
BUSY	OUT	Bool	• BUSY = 1： 作业尚未完成。 无法触发新作业。 • BUSY = 0： 作业已完成。
ERROR	OUT	Bool	状态参数 ERROR： ERROR = 1：处理时出错。 STATUS 提供有关错误类型的详细信息
STATUS	OUT	Word	状态参数 STATUS：错误信息

TRCV



参数	参数类型	数据类型	说明
EN_R	IN	Bool	启用接收的控制参数： EN_R = 1 时，TRCV 准备接收。 正在处理接收作业。
ID	IN	CONN_OUC (Word)	引用相关的连接。标识号必须与本地连接描述中的相关参数标识号相同。 值范围：W#16#0001 到 W#16#0FFF
LEN	IN	Int	接收区长度（字节）（默认值 = 0，这表示 DATA 参数决定要接收的数据的长度）。
DATA	IN_OUT	Variant	指向接收数据的指针： 包含地址和长度的接收区。地址将参考： • 过程映像输入表 • 过程映像输出表 • 位存储器 • 数据块
NDR	OUT	Bool	状态参数 NDR： • NDR = 0： 作业尚未开始或仍在运行。 • NDR = 1： 作业已成功完成。
BUSY	OUT	Bool	• BUSY = 1：作业尚未完成。无法触发新作业。 • BUSY = 0：作业已完成。
ERROR	OUT	Bool	ERROR = 1：处理时出错。STATUS 提供错误类型的详细信息。
STATUS	OUT	Word	错误信息
RCVD_LEN	OUT	Int	实际接收到的数据量（字节）

接收区

TRCV 指令将收到的数据写入到通过以下两个变量指定的接收区：

- 指向区域起始位置的指针
- 区域长度

说明

LEN 参数的默认设置 (LEN = 0) 使用 DATA 参数来确定要传送的数据的长度。 确保 TSEND 指令传送的 DATA 的大小与 TRCV 指令的 DATA 参数的大小相同。

下表说明了 TRCV 如何在接收区输入接收数据。

协议选项	在接收区输入数据	参数连接类型
TCP	指定长度的数据接收	B#16#11
ISO on TCP	协议控制	B#16#12

接收所有作业数据后，TRCV 立即将其传送到接收区并将 NDR 设置为 1。

TCON 的条件代码

ERROR	STATUS (W#16#...)	说明
0	0000	连接已成功建立
0	7000	无激活的作业处理
0	7001	启动作业处理，正在建立连接
0	7002	后续调用（与 REQ 不相关），正在建立连接
1	8086	ID 参数超出允许范围。
1	8087	已达到最大连接数；无法建立更多连接
1	809B	连接描述中的 local_device_id 与 CPU 的不匹配。
1	80A1	连接或端口已被用户占用
1	80A2	本地端口或远程端口已被系统占用
1	80A3	正尝试重新建立现有连接
1	80A4	远程连接端点的 IP 地址无效；可能与本地 IP 地址匹配

ERROR	STATUS (W#16#...)	说明
1	80A7	通信错误： 在 TCON 完成前执行了 TDISCON。 TDISCON 必须先完全终止 ID 引用的连接。
1	80B3	不一致的参数分配： 错误代码 W#16#80A0 到 W#16#80A2、W#16#80A4、W#16#80B4 到 W#16#80B9 对应的组错误
1	80B4	使用 ISO on TCP (connection_type = B#16#12) 建立被动连接时，条件代码 80B4 提示您输入的 TSAP 不符合下列某一项地址要求： • 若是本地 TSAP 长度为 2 个字节且首字节的 TSAP ID 值为 E0 或 E1（十六进制），第二字节必须为 00 或 01。 • 如果本地 TSAP 长度为 3 个或更多字节，且首字节的 TSAP ID 值为 E0 或 E1（十六进制），则第二字节必须为 00 或 01，且所有其它字节必须为有效的 ASCII 字符。 • 如果本地 TSAP 长度为 3 个或更多字节，且首字节的 TSAP ID 值既不为 E0 也不为 E1（十六进制），则 TSAP ID 的所有字节都必须为有效的 ASCII 字符。 有效 ASCII 字符的字节值为 20 到 7E（十六进制）。
1	80B5	active_est 参数错误
1	80B6	参数 connection_type 的参数分配错误
1	80B7	以下参数之一有错误： block_length、local_tsap_id_len、rem_subnet_id_len、rem_staddr_len、rem_tsap_id_len、next_staddr_len
1	80B8	本地连接描述中的参数与参数 ID 不同
1	80C3	所有连接资源都在使用。
1	80C4	临时通信错误： • 此时无法建立连接。 • 接口正在接收新参数。 • TDISCON 当前正在删除已组态连接。

TDISCON 的条件代码

ERROR	STATUS (W#16#...)	说明
0	0000	连接已成功终止
0	7000	无激活的作业处理
0	7001	启动作业处理，正在终止连接
0	7002	后续调用（与 REQ 不相关），正在终止连接
1	8086	ID 参数不在允许的地址范围内。
1	80A3	尝试终止的连接不存在
1	80C4	临时通信错误： 接口正在接收新参数或当前正在建立连接。

TSEND 的条件代码

ERROR	STATUS (W#16#...)	说明
0	0000	发送作业无错完成
0	7000	无激活的作业处理
0	7001	启动作业处理，正在发送数据： 在执行此处理期间，操作系统访问 DATA 发送区中的数据。
0	7002	后续调用（与 REQ 不相关），正在处理作业： 在执行此处理期间，操作系统访问 DATA 发送区中的数据。
1	8085	LEN 参数的值比最大的允许值大。
1	8086	ID 参数不在允许的地址范围内
1	8088	LEN 参数大于在 DATA 中指定的存储区
1	80A1	通信错误： • 尚未建立指定的连接 • 当前正在终止指定的连接。 无法通过此连接进行传送。 • 正在重新初始化接口。

ERROR	STATUS (W#16#...)	说明
1	80C3	内部缺乏资源： 已经在以其它优先等级处理具有该 ID 的块。
1	80C4	临时通信错误： • 此时无法建立与通信伙伴的连接。 • 接口正在接收新参数或当前正在建立连接。

TRCV 的条件代码

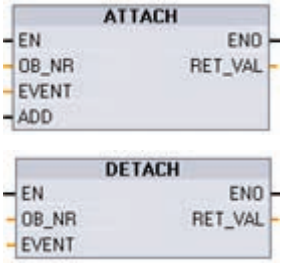
ERROR	STATUS (W#16#...)	说明
0	0000	已接受新数据： 在 RCVD_LEN 中显示已接收数据的当前长度。
0	7000	块未准备好接收
0	7001	块准备接收，接收作业已激活。
0	7002	后续调用，正在处理接收作业： 数据在执行此处理期间写入接收区。因此，错误可能会导致接收区中的数据不一致。
1	8085	LEN 参数值大于最大允许值，或者自第一次调用以来更改了 LEN 或 DATA 参数。
1	8086	ID 参数不在允许的地址范围内
1	8088	接收区过小： LEN 的值大于 DATA 指定的接收区。
1	80A1	通信错误： • 尚未建立指定的连接 • 当前正在终止指定的连接。 无法通过该连接执行接收作业。 • 接口正在接收新参数。
1	80C3	内部缺乏资源：已经在以其它优先等级处理具有该 ID 的块。
1	80C4	临时通信错误： • 此时无法建立与伙伴的连接。 • 接口正在接收新参数设置或当前正在建立连接。

6.2.5.2 点对点指令

点对点 (PtP) 一章 (页 273)提供了有关 PtP 指令和通信模块的详细信息。

6.2.6 中断指令

6.2.6.1 附加和分离指令



使用 ATTACH 和 DETACH 指令可激活和禁用中断事件驱动的子程序。

- ATTACH 启用响应硬件中断事件的中断 OB 子程序执行。
- DETACH 禁用响应硬件中断事件的中断 OB 子程序执行。

参数	参数类型	数据类型	说明
OB_NR	IN	Int	组织块标识符： 从使用“添加新块”(Add new block) 功能创建的可用硬件中断 OB 中进行选择。 双击该参数域，然后单击助手图标可查看可用的 OB。
EVENT	IN	DWord	事件标识符： 从在 PLC 设备配置中为数字输入或高速计数器启用的可用硬件中断事件中进行选择。 双击该参数域，然后单击助手图标可查看这些可用事件。
ADD（仅 ATTACH）	IN	Bool	ADD = 0（默认）： 该事件将取代先前为此 OB 附加的所有事件。 ADD = 1： 该事件将添加到先前为此 OB 附加的事件中。
RET_VAL	OUT	Int	执行条件代码

硬件中断事件

CPU 支持以下硬件中断事件：

- 上升沿事件（所有内置 CPU 数字量输入外加任何信号板数字量输入）
  - 数字输入从 OFF 切换为 ON 时会出现上升沿，以响应连接到输入的现场设备的信号变化。
- 下降沿事件（所有内置 CPU 数字量输入外加任何信号板输入）
  - 数字输入从 ON 切换为 OFF 时会出现下降沿。
- 高速计数器 (HSC) 当前值 = 参考值 (CV = RV) 事件（HSC 1 至 6）
  - 当前计数值从相邻值变为与先前设置的参考值完全匹配时，会生成 HSC 的 CV = RV 中断。
- HSC 方向变化事件（HSC 1 至 6）
  - 当检测到 HSC 从增大变为减小或从减小变为增大时，会发生方向变化事件。
- HSC 外部复位事件（HSC 1 至 6）
  - 某些 HSC 模式允许分配一个数字输入作为外部复位端，用于将 HSC 的计数值重置为零。 当该输入从 OFF 切换为 ON 时，会发生此类 HSC 的外部复位事件。

在设备配置期间启用硬件中断事件

必须在设备配置中启用硬件中断。 如果要在配置或运行期间附加此事件，则必须在设备配置中为数字输入通道或 HSC 选中启用事件框。

PLC 设备配置中的复选框选项：

- 数字输入
  - 启用上升沿检测
  - 启用下降沿检测
- 高速计数器 (HSC)
  - 启用此高速计数器
  - 生成计数器值等于参考计数值的中断
  - 生成外部复位事件的中断
  - 生成方向变化事件的中断

向用户程序添加新硬件中断 OB 代码块

默认情况下，第一次启用事件时，没有任何 OB 附加到该事件。 这会通过“HW 中断：”(HW interrupt:) 设备配置“<未连接>”(<not connected>) 标签来指示。 只有硬件中断 OB 能附加到硬件中断事件。 所有现有的硬件中断 OB 都会出现在“HW 中断：”(HW interrupt:) 下拉列表中。 如果未列出任何 OB，则必须按下列步骤创建类型为“硬件中断”的 OB。 在项目树的“程序块”(Program blocks) 分支下：

1. 双击“添加新块”(Add new block)，选择“组织块 (OB)”(Organization block (OB))，然后选择“硬件中断”(Hardware interrupt)。
2. 也可以重命名 OB、选择编程语言（LAD 或 FBD）以及选择块编号（切换为手动并选择与建议块编号不同的块编号）。
3. 编辑该 OB，添加事件发生时要执行的已编程响应。 可以从此 OB 调用最多嵌套四层深的 FC 和 FB。

OB_NR 参数

所有现有的硬件中断 OB 名称都会出现在设备配置“HW 中断：”(HW interrupt:) 下拉列表和 ATTACH/DETACH 参数 OB_NR 下拉列表中。

EVENT 参数

启用某个硬件中断事件时，将为该事件分配一个唯一的默认事件名称。 可通过编辑“事件名称：”(Event name:) 编辑框更改此事件名称，但必须是唯一的名称。 这些事件名称将成为“常量”(Constants) 变量表中的变量名称，并出现在 ATTACH 和 DETACH 指令框的 EVENT 参数下拉列表中。 变量的值是用于标识事件的内部编号。

常规操作

每个硬件事件都可附加到一个硬件中断 OB 中，在发生该硬件中断事件时将排队执行该硬件中断 OB。 在组态或运行期间可附加 OB 事件。

用户可以在组态时将 OB 附加到已启用的事件或使其与该事件分离。 要在组态时将 OB 附加到事件，必须使用“HW 中断：”(HW interrupt:) 下拉列表（单击右侧的向下箭头）并从可用硬件中断 OB 的列表中选择 OB。 从该列表中选择相应的 OB 名称，或者选择“<未连接>”(<not connected>) 以删除该附加关系。

也可以在运行期间附加或分离已启用的硬件中断事件。 在运行期间使用 ATTACH 或 DETACH 程序指令（如有必要可多次使用）将已启用的中断事件附加到相应的 OB 或与其分离。如果当前未附加到任何 OB（选择了设备配置中的“<未连接>”(<not connected>) 选项或由于执行了 DETACH 指令），则将忽略已启用的硬件中断事件。

DETACH 操作

使用 DETACH 指令将特定事件或所有事件与特定 OB 分离。如果指定了 EVENT，则仅将该事件与指定的 OB_NR 分离；当前附加到此 OB_NR 的任何其它事件仍保持附加状态。 如果未指定 EVENT，则分离当前连接到 OB_NR 的所有事件。

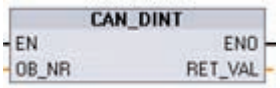
条件代码

RET_VAL (W#16#....)	ENO 状态	说明
0000	1	无错误
0001	0	没有要分离的事件（仅 DETACH）
8090	0	OB 不存在
8091	0	OB 类型错误
8093	0	事件不存在

6.2.6.2 启动和取消延时中断指令

通过 SRT_DINT 和 CAN_DINT 指令可以启动和取消延时中断处理过程。 每个延时中断都是一个在指定的延迟时间过后发生的一次性事件。 如果在延迟时间到期前取消延时事件，则不会发生程序中断。

参数 DTIME 指定的延迟时间过去后，SRT_DINT 会启动执行 OB（组织块）子程序的延时中断。



CAN_DINT 可取消已启动的延时中断。 在这种情况下，将不执行延时中断 OB。

SRT_DINT 参数

参数	参数类型	数据类型	说明
OB_NR	IN	Int	将在延迟时间过后启动的组织块 (OB)： 从使用“添加新块”(Add new block) 项目树功能创建的可用延时中断 OB 中进行选择。 双击该参数域，然后单击助手图标可查看可用的 OB。
DTIME	IN	Time	延迟时间值（1 到 60000 ms） 可创建更长的延迟时间，例如，可以通过在延时中断 OB 内使用计数器来实现。
SIGN	IN	Word	未被 S7-1200 使用；任何值都接受
RET_VAL	OUT	Int	执行条件代码

CAN_DINT 参数

参数	参数类型	数据类型	说明
OB_NR	IN	Int	延时中断 OB 标识符。 可使用 OB 编号或符号名称。
RET_VAL	OUT	Int	执行条件代码

操作

SRT_DINT 指令指定延迟时间、启动内部延迟时间定时器以及将延时中断 OB 子程序与延时超时事件相关联。 指定的延迟时间过去后，将生成可触发相关延时中断 OB 执行的程序中断。在指定的延时发生之前执行 CAN_DINT 指令可取消进行中的延时中断。 激活延时和时间循环中断事件的总次数不得超过四次。

在项目中添加延时中断 OB 子程序

只有延时中断 OB 可分配给 SRT_DINT 和 CAN_DINT 指令。 新项目中不存在延时中断 OB。 必须将延时中断 OB 添加到项目中。 要创建延时中断 OB，请按以下步骤操作：

1. 在项目树的“程序块”(Program blocks) 分支中双击“添加新块”(Add new block)，选择“组织块 (OB)”(Organization block (OB))，然后选择“延时中断”(Time delay interrupt)。
2. 可以重命名 OB、选择编程语言或选择块编号。 如果要分配与自动分配的编号不同的块编号，请切换到手动编号模式。

3. 编辑延时中断 OB 子程序，并创建要在发生延时超时事件时执行的已编程响应。 可从延时中断 OB 调用其它最多嵌套四层深的 FC 和 FB 代码块。
4. 编辑 SRT_DINT 和 CAN_DINT 指令的 OB_NR 参数时，将可以使用新分配的延时中断 OB 名称。

条件代码

RET_VAL (W#16#...)	说明
0000	未出错
8090	不正确的参数 OB_NR
8091	不正确的参数 DTIME
80A0	未启动延时中断

6.2.6.3 禁用和启用报警中断指令

使用 DIS_AIRT 和 EN_AIRT 指令可禁用和启用报警中断处理过程。

DIS_AIRT 可延迟新中断事件的处理。 您可在 OB 中多次执行 DIS_AIRT。DIS_AIRT 执行次数由操作系统进行计数。 在特别通过 EN_AIRT 指令再次取消之前或者在已完成处理当前 OB 之前，这些执行中的每一个都保持有效。

再次启用这些执行后，将立即处理 DIS_AIRT 生效期间发生的中断，或者在完成执行当前 OB 后，立即处理中断。

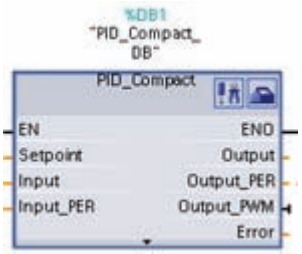
对先前使用 DIS_AIRT 指令禁用的中断事件处理，可使用 EN_AIRT 来启用。 每一次 DIS_AIRT 执行都必须通过一次 EN_AIRT 执行来取消。 例如，如果通过五次 DIS_AIRT 执行禁用中断五次，则必须通过五次 EN_AIRT 执行来取消。

必须在同一个 OB 中或从同一个 OB 调用的任意 FC 或 FB 中完成 EN_AIRT 执行后，才能再次启用此 OB 的中断。

参数 RET_VAL 表示禁用中断处理的次数，即已排队的 DIS_AIRT 执行的个数。 只有当参数 RET_VAL = 0 时，才会再次启用中断处理。

参数	参数类型	数据类型	说明
RET_VAL	OUT	Int	延迟次数 = 队列中的 DIS_AIRT 执行次数。

6.2.7 PID 控制



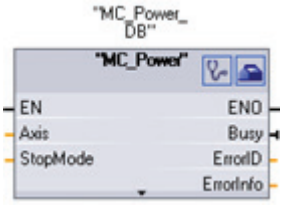
“PID_Compact”语句可用来提供可在自动和手动模式下自我优化调节的 PID 控制器。

有关 PID_Compact 指令的信息，请参见 TIA 门户的在线帮助。

6.2.8 运动控制指令

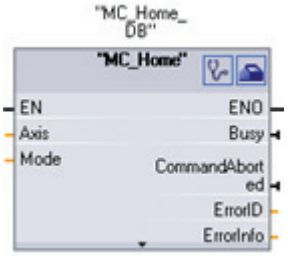
运动控制指令使用相关工艺数据块和 CPU 的专用 PTO（Pulse Train Outputs，脉冲串输出）来控制轴上的运动。 有关运动控制指令的信息，请参见 STEP 7 Basic 的在线帮助。

注意
脉冲输出发生器的最大脉冲频率对于 CPU 的数字量输出为 100 KHz，而对于信号板的数字量输出为 20 KHz。可是，当组态了最大速度或频率超出此硬件限制的轴时，STEP 7 Basic 并不会提醒用户。这可能会导致应用出现问题，因此请始终确保不会超出硬件的最大脉冲频率。



MC_Power 可启用和禁用运动控制轴。

MC_Reset 可复位所有运动控制错误。 所有可确认的运动控制错误都会被确认。



MC_Home 可建立轴控制程序与轴机械定位系统之间的关系。

MC_Halt 可取消所有运动过程并使轴运动停止。 停止位置未定义。

MC_MoveJog 可执行用于测试和启动目的的点动模式。

MC_MoveAbsolute 可启动到某个绝对位置的运动。 该作业在到达目标位置时结束。

MC_MoveRelative 可启动相对于起始位置的定位运动。

MC_MoveVelocity 可使轴以指定的速度平动。

说明

用户程序中的其它指令无法使用脉冲串输出

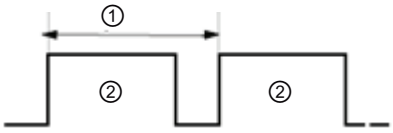
将 CPU 或信号板的输出组态为脉冲发生器时（供 PWM 或基本运动控制指令使用），这会从 Q 存储器中移除相应的输出地址（Q0.0、Q0.1、Q4.0 和 Q4.1），并且这些地址在用户程序中不能用于其它用途。如果用户程序向用作脉冲发生器的输出写入值，则 CPU 不会将该值写入到物理输出。

6.2.9 脉冲指令

6.2.9.1 CTRL_PWM 指令

CTRL_PWM 脉冲宽度调制 (PWM, Pulse Width Modulation) 指令可提供占空比可变的固定循环时间输出。 PWM 输出以指定频率（循环时间）启动之后将连续运行。

脉冲宽度会根据需要进行变化以影响所需的控制。



① 循环时间

② 脉冲宽度

脉冲宽度可表示为循环时间的百分数 (0 - 100)、千分数 (0 - 1000)、万分数 (0 - 10000) 或 S7 模拟格式。 脉冲宽度可从 0（无脉冲，始终关闭）到满刻度（无脉冲，始终打开）变化。

由于 PWM 输出可从 0 到满刻度变化，因此可提供在许多方面都与模拟输出相同的数字输出。 例如，PWM 输出可用于控制电机的速度，速度范围可以是从停止到全速；也可用于控制阀的位置，位置范围可以是从闭合到完全打开。

有两种脉冲发生器可用于控制高速脉冲输出功能： PWM 和脉冲串输出 (PTO, Pulse train output)。 PTO 由运动控制指令使用。 可将每个脉冲发生器指定为 PWM 或 PTO，但不能指定为既是 PWM 又是 PTO。

这两种脉冲发生器映射到特定的数字输出，如下表所示。 可以使用板载 CPU 输出，也可以使用可选的信号板输出。 下表列出了输出点编号（假定使用默认输出组态）。 如果更改了输出点编号，则输出点编号将为用户指定的编号。 无论是在 CPU 上还是在连接的信号板上，PTO1/PWM1 都使用前两个数字输出，PTO2/PWM2 使用接下来的两个数字输出。 请注意，PWM 仅需要一个输出，而 PTO 每个通道可选择使用两个输出。 如果脉冲功能不需要输出，则相应的输出可用于其它用途。

说明	默认输出分配		
		脉冲	方向
PTO 1	板载 CPU	Q0.0	Q0.1
	信号板	Q4.0	Q4.1
PWM 1	板载 CPU	Q0.0	--
	信号板	Q4.0	--
PTO 2	板载 CPU	Q0.2	Q0.3
	信号板	Q4.2	Q4.3
PWM 2	板载 CPU	Q0.2	--
	信号板	Q4.2	--

组态 PWM 的脉冲通道

要准备 PWM 操作，首先通过选择 CPU 来组态设备配置中的脉冲通道，然后组态脉冲发生器 (PTO/PWM)，并选择 PWM1 或 PWM2。 启用脉冲发生器（复选框）。 如果启用一个脉冲发生器，将为该特定脉冲发生器分配一个唯一的默认名称。 可编辑“名称：”(Name:) 编辑框中的名称来更改名称，但必须是唯一的名称。 已启用的脉冲发生器的名称将成为“常量”(constant) 变量表中的变量，并可用作 CTRL_PWM 指令的 PWM 参数。

注意
脉冲输出发生器的最大脉冲频率对于 CPU 的数字量输出为 100 KHz，而对于信号板的数字量输出为 20 KHz。可是，当组态了最大速度或频率超出此硬件限制的轴时，STEP 7 Basic 并不会提醒用户。这可能会导致应用出现问题，因此请始终确保不会超出硬件的最大脉冲频率。

可按如下方式重命名脉冲发生器、添加注释以及分配参数：

- 脉冲发生器可用作： PWM 或 PTO（选择 PWM）
- 输出源： 板载 CPU 或信号板
- 时间基数： 毫秒或微秒

- 脉冲宽度格式：
  - 百分数（0 到 100）
  - 千分数（0 到 1000）
  - 万分数（0 到 10000）
  - S7 模拟格式（0 到 27648）
- 循环时间： 输入循环时间值。 该值只能在“设备配置”(Device configuration) 中更改。
- 初始脉冲宽度： 输入初始脉冲宽度值。 可在运行期间更改脉冲宽度值。

输出地址

起始地址： 输入要在其中查找脉冲宽度值的 Q 字地址。 对于 PWM1，默认位置是 QW1000；而对于 PWM2，默认位置是 QW1002。 该位置的值控制脉冲宽度，并且在每次 CPU 从 STOP 切换到 RUN 模式时都会初始化为上面指定的“初始脉冲宽度：”(Initial pulse width:) 值。 在运行期间更改该 Q 字值会引起脉冲宽度变化。

参数	参数类型	数据类型	初始值	说明
PWM	IN	Word	0	PWM 标识符： 已启用的脉冲发生器的名称将变为“常量”(constant) 变量表中的变量，并可用作 PWM 参数。
ENABLE	IN	Bool		1 = 启动脉冲发生器 0 = 停止脉冲发生器
BUSY	OUT	Bool	0	功能忙
STATUS	OUT	Word	0	执行条件代码

操作

CTRL_PWM 指令使用数据块 (DB) 来存储参数信息。 在程序编辑器中放置 CTRL_PWM 指令时，将分配 DB。 数据块参数不是由用户单独更改的，而是由 CTRL_PWM 指令进行控制。

通过将其变量名称用于 PWM 参数，指定要使用的已启用脉冲发生器。

EN 输入为 TRUE 时，PWM_CTRL 指令根据 ENABLE 输入的值启动或停止所标识的 PWM。 脉冲宽度由相关 Q 字输出地址中的值指定。

由于 S7-1200 在 CTRL_PWM 指令执行后处理请求，所以在 S7-1200 CPU 型号上，参数 BUSY 总是报告 FALSE。

如果检测到错误，则 ENO 设置为 FALSE 且参数 STATUS 包含条件代码。

PLC 第一次进入 RUN 模式时，脉冲宽度将设置为在设备配置中组态的初始值。 根据需要将值写入设备配置中指定的 Q 字位置（“输出地址”/“起始地址：”）以更改脉冲宽度。使用指令（如移动、转换、数学）或 PID 功能框将所需脉冲宽度写入相应的 Q 字。 必须使用 Q 字值的有效范围（百分数、千分数、万分数或 S7 模拟格式）。

条件代码

STATUS 值	说明
0	无错误
80A1	PWM 标识符未寻址到有效的 PWM

无法强制分配给 PWM 和 PTO 的数字量 I/O 点

在设备配置期间分配脉冲宽度调制 (PWM, Pulse-Width Modulation) 和脉冲串输出 (PTO, Pulse-Train Output) 设备使用的数字量 I/O 点。将数字 I/O 点分配给这些设备之后，无法通过监视表格强制功能修改所分配的 I/O 点的地址值。

用户程序中的其它指令无法使用脉冲串输出

将 CPU 或信号板的输出组态为脉冲发生器时（供 PWM 或基本运动控制指令使用），这会从 Q 存储器中移除相应的输出地址（Q0.0、Q0.1、Q4.0 和 Q4.1），并且这些地址在用户程序中不能用于其它用途。如果用户程序向用作脉冲发生器的输出写入值，则 CPU 不会将该值写入到物理输出。

6.3 全局库指令

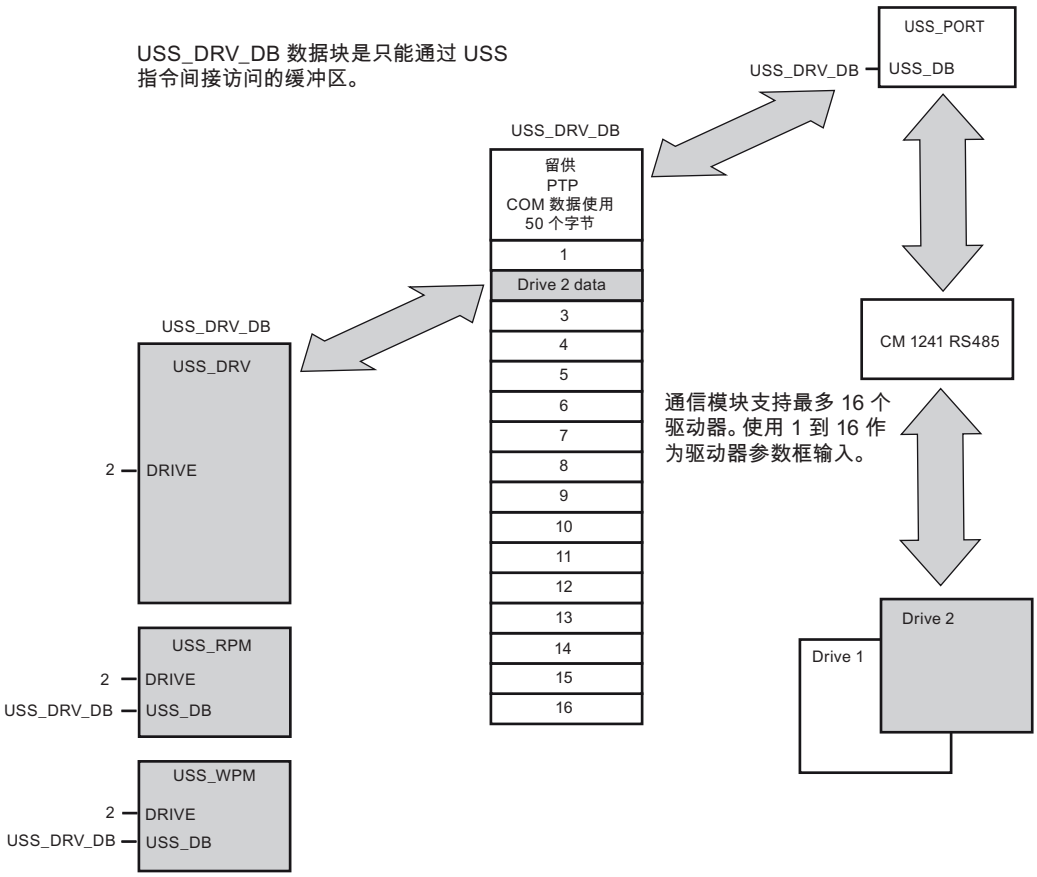
6.3.1 USS

USS 协议库可控制支持 USS 协议的西门子驱动器。 这些指令包括专为使用 USS 协议与驱动器进行通信而设计的功能。 CM 1241 RS485 模块通过 RS485 端口与驱动器进行通信。 可使用 USS 库控制物理驱动器和读/写驱动器参数。

6.3.1.1 使用 USS 协议的要求

该库提供 1 个 FB 和 3 个 FC 来支持 USS 协议。 每个 CM1241 RS485 通信模块最多支持 16 个驱动器。

对于与所安装的各个 PtP 通信模块相连接的 USS 网络，在单个背景数据块中包含用于该网络中所有驱动器的临时存储区和缓冲区。 这些驱动器的 USS 功能共享该数据块中的信息。



连接到一个 CM 1241 RS485 的所有驱动器（最多 16 个）是同一 USS 网络的一部分。 连接到另一 CM 1241 RS485 的所有驱动器是另一 USS 网络的一部分。 因为 S7-1200 最多支持三个 CM 1241 RS485 设备，所以用户最多可建立三个 USS 网络，每个网络最多 16 个驱动器，总共支持 48 个 USS 驱动器。

各 USS 网络使用唯一的数据块进行管理（使用三个 CM 1241 RS485 设备建立三个 USS 网络需要三个数据块）。 与各 USS 网络相关的所有指令必须共享该数据块。 这包括用于控制各 USS 网络上的所有控制器的所有 USS_DRV、USS_PORT、USS_RPM 和 USS_WPM 指令。

USS_DRV 指令是功能块 (FB)。 在编辑器中放置 USS_DRV 指令时，系统将通过“调用选项”(Call options) 对话框提示您为该 FB 分配哪个 DB。 如果对于该 USS 网络而言，它是该程序中的第一条 USS_DRV 指令，则可以接受默认的 DB 分配（或根据需要更改名称），将相应地创建一个新的 DB。 但如果对于该通道它不是第一条 USS_DRV 指令，则必须使用“调用选项”(Call options) 对话框中的下拉列表选择先前为该 USS 网络分配的相应 DB。

指令 USS_PORT、USS_RPM 和 USS_WPM 全部都是功能 (FC)。 在编辑器中放置这些 FC 时不分配 DB。 您必须改为亲自将合适的 DB 分配给这些指令的“USS_DB”输入（双击该参数域，然后单击助手图标可查看可用的 DB）。

USS_PORT 功能通过 PtP 通信模块处理 CPU 和驱动器之间的实际通信。 每次调用此功能可处理与一个驱动器的一次通信。 用户程序必须尽快调用此功能以防止与驱动器通信超时。 可在主 OB 或任何中断 OB 中调用此功能。

用户程序通过 USS_DRV 功能块可访问 USS 网络上指定的驱动器。 其输入和输出是驱动器的状态和控制。 如果网络上有 16 个驱动器，则用户程序必须具有至少 16 个 USS_DRV 调用，每个驱动器一个调用。 应该以控制驱动器功能所需的速率调用这些块。

只能从主 OB 中调用 USS_DRV 功能块。

⚠小心
只能从主 OB 中调用 USS_DRV、USS_RPM 和 USS_WPM。可从任何 OB 中调用 USS_PORT 功能，通常从延时中断中调用。 未能防止 USS_PORT 中断可能会产生意外错误。

USS_RPM 和 USS_WPM 功能可读取和写入远程驱动器工作参数。 这些参数控制驱动器的内部运行。 有关这些参数的定义，请参见驱动器手册。 用户程序可包含尽可能多的这些功能，但在任何特定时刻，每个驱动器只能激活一个读或写请求。 只能从主 OB 中调用 USS_RPM 和 USS_WPM 功能。

计算与驱动器通信所需的时间

与驱动器进行的通信与 S7-1200 扫描不同步。 在完成一个驱动器通信事务之前，S7-1200 通常完成了多个扫描。

USS_PORT 间隔是一个驱动器事务所需的时间。 下表列出了各个波特率的最小 USS_PORT 间隔。 比 USS_PORT 间隔更频繁地调用 USS_PORT 功能不会增加事务数。 如果通信错误导致尝试 3 次才能完成事务，则驱动器超时间隔是处理该事务可能花费的时间。 默认情况下，USS 协议库对每个事务最多自动进行 2 次重试。

波特率	计算的最小 USS_PORT 调用间隔（毫秒）	每个驱动器的驱动器消息间隔超时（毫秒）
1200	790	2370
2400	405	1215
4800	212.5	638
9600	116.3	349
19200	68.2	205
38400	44.1	133
57600	36.1	109
115200	28.1	85

6.3.1.2 USS_DRV 指令

USS_DRV 指令通过创建请求消息和解释驱动器响应消息与驱动器交换数据。 每个驱动器都应使用单独的一个功能块，但是与一个 USS 网络和 PtP 通信模块相关的所有 USS 功能都必须使用同一个背景数据块。 必须在放置第一个 USS_DRV 指令时创建该 DB 名称，然后可重复使用通过该初始指令使用而创建的这个 DB。

首次执行 USS_DRV 时，将在背景数据块中初始化由 USS 地址（参数 DRIVE）指示的驱动器。 完成初始化后，随后执行 USS_PORT 即可开始与具有此驱动器编号的驱动器通信。

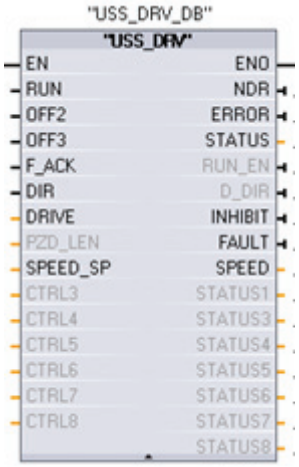
更改驱动器编号操作将要求 PLC 从 STOP 模式切换到 RUN 模式以初始化相应的背景 DB。 将输入参数组态到 USS TX 消息缓冲区中，并从“前一个”有效响应缓冲区（如果存在）读取输出。 USS_DRV 执行期间没有数据传输。 执行 USS_PORT 后即可与驱动器进行通信。 USS_DRV 仅组态要发送的消息并解释已从前一个请求中接收的数据。

用户可以同时使用 DIR 输入 (BOOL) 或符号（正或负）和 SPEED_SP 输入 (REAL) 控制驱动器旋转方向。 下表假定电机按正向旋转接线，说明这些输入如何一起决定驱动器旋转方向。

SPEED_SP	DIR	驱动器旋转方向
数值 > 0	0	反转
数值 > 0	1	正转
数值 < 0	0	正转
数值 < 0	1	反转

LAD（默认视图）

LAD（扩展视图）



通过单击功能框的底部展开该功能框可显示所有参数。

灰显的参数引脚可选，不需要进行分配。

参数	参数类型	数据类型	说明
RUN	IN	Bool	驱动器起始位： 该输入为真时，将使驱动器以预设速度运行。
OFF2	IN	Bool	电气停止位： 该位为假时，将使驱动器在不经过制动的情况下逐渐自然停止。
OFF3	IN	Bool	快速停止位 – 该位为假时，将通过制动使驱动器快速停止，而不只是使驱动器逐渐自然停止。

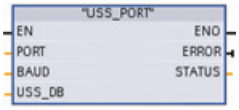
参数	参数类型	数据类型	说明
F_ACK	IN	Bool	故障确认位 – 设置该位以复位驱动器上的故障位。 清除故障后会设置该位，以告知驱动器不再需要指示前一个故障。
DIR	IN	Bool	驱动器方向控制 – 设置该位以指示方向为向前（对于正 SPEED_SP）。
DRIVE	IN	USInt	驱动器地址：该输入是 USS 驱动器的地址。 有效范围是驱动器 1 到驱动器 16。
PZD_LEN	IN	USInt	字长度 – 这是 PZD 数据的字数。 有效值为 2、4、6 或 8 个字。 默认值为 2。
SPEED_SP	IN	Real	速度设定值 – 这是以组态频率的百分数形式表示的驱动器速度。 正值表示方向向前（DIR 为真时）。
CTRL3	IN	UInt	控制字 3 – 写入驱动器上用户可组态参数的值。用户必须在驱动器上组态该值。 可选参数。
CTRL4	IN	UInt	控制字 4 – 写入驱动器上用户可组态参数的值。用户必须在驱动器上组态该值。 可选参数。
CTRL5	IN	UInt	控制字 5 – 写入驱动器上用户可组态参数的值。用户必须在驱动器上组态该值。 可选参数。
CTRL6	IN	UInt	控制字 6 – 写入驱动器上用户可组态参数的值。用户必须在驱动器上组态该值。
CTRL7	IN	UInt	控制字 7 – 写入驱动器上用户可组态参数的值。用户必须在驱动器上组态该值。 可选参数。
CTRL8	IN	UInt	控制字 8 – 写入驱动器上用户可组态参数的值。用户必须在驱动器上组态该值。 可选参数。
NDR	OUT	Bool	新数据就绪 – 该位为真时，表示输出包含新通信请求数据。
ERROR	OUT	Bool	发生错误 – 该参数为真时，表示发生错误，STATUS 输出有效。其它所有输出在出错时均设置为零。 仅在 USS_PORT 指令的 ERROR 和 STATUS 输出中报告通信错误。
STATUS	OUT	UInt	请求的状态值。 它指示扫描的结果。 这不是从驱动器返回的状态字。
RUN_EN	OUT	Bool	运行已启用 – 该位指示驱动器是否在运行。

参数	参数类型	数据类型	说明
D_DIR	OUT	Bool	驱动器方向 – 该位指示驱动器是否在向前运行。
INHIBIT	OUT	Bool	驱动器已禁止 – 该位指示驱动器上禁止位的状态。
FAULT	OUT	Bool	驱动器故障 – 该位指示驱动器已注册故障。 用户必须解决问题，并且在该位被置位时，设置 F_ACK 位以清除此位。
SPEED	OUT	REAL	驱动器当前速度（驱动器状态字 2 的标定值）– 以组态速度的百分数表示的驱动器速度值。
STATUS1	OUT	UInt	驱动器状态字 1 – 该值包含驱动器的固定状态位。
STATUS3	OUT	UInt	驱动器状态字 3 – 该值包含驱动器上用户可组态的状态字。
STATUS4	OUT	UInt	驱动器状态字 4 – 该值包含驱动器上用户可组态的状态字。
STATUS5	OUT	UInt	驱动器状态字 5 – 该值包含驱动器上用户可组态的状态字。
STATUS6	OUT	UInt	驱动器状态字 6 – 该值包含驱动器上用户可组态的状态字。
STATUS7	OUT	UInt	驱动器状态字 7 – 该值包含驱动器上用户可组态的状态字。
STATUS8	OUT	UInt	驱动器状态字 8 – 该值包含驱动器上用户可组态的状态字。

6.3.1.3 USS_PORT 指令

USS_PORT 指令用于处理 USS 网络上的通信。 通常程序中每个 PtP 通信模块只一个 USS_PORT 功能，且每次调用该功能都会处理与单个驱动器的通信。 用户程序必须尽快执行 USS_PORT 功能以防止驱动器超时。 与同一个 USS 网络和 PtP 通信模块相关的所有 USS 功能必须使用同一个背景数据块。 通常从延时中断 OB 调用 USS_PORT 以防止驱动器超时以及使可供 USS_DRV 调用的 USS 数据保持最新。

LAD

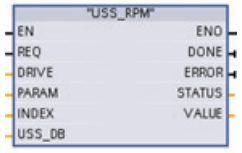


FBD

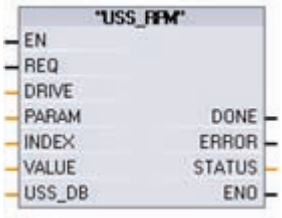
参数	参数类型	数据类型	说明
PORT	IN	端口	PtP 通信模块。 标识符： 这是可在默认变量表的“常量”(Constants) 选项卡内引用的常量。
BAUD	IN	DInt	USS 通信要使用的波特率。
USS_DB	IN	DInt	这是对在用户程序中放置 USS_DRV 指令时创建和初始化的背景数据块的引用。
ERROR	OUT	Bool	该引脚为真时，表示发生错误，STATUS 输出有效。
STATUS	OUT	UInt	请求的状态值。 它指示扫描或初始化的结果。 对于有些状态代码，还在“USS_Extended_Error”变量中提供了更多信息。

6.3.1.4 USS_RPM 指令

LAD



FBD



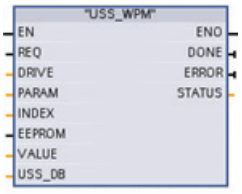
USS_RPM 指令用于从驱动器读取参数。与同一个 USS 网络和 PtP 通信模块相关的所有 USS 功能必须使用同一个数据块。必须从主 OB 中调用 USS_RPM。

参数	参数类型	数据类型	说明
REQ	IN	Bool	发送请求： 该参数为真时，表示需要新的读请求。如果该参数的请求已处于待决状态，将忽略新请求。
DRIVE	IN	USInt	驱动器地址：该输入是 USS 驱动器的地址。有效范围是驱动器 1 到驱动器 16。
PARAM	IN	UInt	参数编号：此输入指示要写入的驱动器参数。该参数的范围为 0 到 2047。有关如何访问超出该范围的任何参数的详细信息，请参见驱动器手册。
INDEX	IN	UInt	参数索引：该输入指示要写入的驱动器参数索引。索引为一个 16 位值，其中最低有效字节是实际索引值，其范围是 0 到 255。最高有效字节也可被驱动器使用且取决于驱动器。 有关详细信息，请参见驱动器手册。
USS_DB	IN	Variant	这是对在用户程序中放置 USS_DRV 指令时创建和初始化的背景数据块的引用。
VALUE	IN	Word、Int、UInt、DWord、DInt、UDInt、Real	这是已读取的参数的值，仅当 DONE 位为真时才有效。

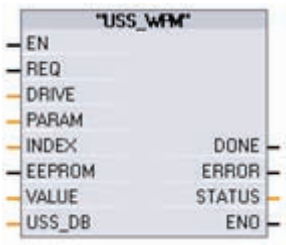
参数	参数类型	数据类型	说明
DONE	OUT	Bool	完成： 该参数为真时，表示 VALUE 输出包含先前请求的读取参数值。 USS_DRV 发现来自驱动器的读响应数据时会设置该位。 满足以下条件之一时复位该位： • 用户通过另一个 USS_RPM 轮询请求响应数据 或 • 执行接下来两个 USS_DRV 调用的第二个调用时
ERROR	OUT	Bool	发生错误 – 该参数为真时，表示发生错误，STATUS 输出有效。其它所有输出在出错时均设置为零。 仅在 USS_PORT 指令的 ERROR 和 STATUS 输出中报告通信错误。
STATUS	OUT	UInt	这是请求的状态值。 它表示读请求的结果。 对于有些状态代码，还在“USS_Extended_Error”变量中提供了更多信息。

6.3.1.5 USS_WPM 指令

LAD



FBD



USS_WPM 指令用于修改驱动器中的参数。与同一个 USS 网络和 PtP 通信模块相关的所有 USS 功能必须使用同一个数据块。必须从主 OB 中调用 USS_WPM。

说明

EEPROM 写操作

注意不要过多使用 EEPROM 永久写操作。 请尽可能减少 EEPROM 写操作次数以延长 EEPROM 的寿命。

参数	参数类型	数据类型	说明
REQ	IN	Bool	发送请求： 该参数为真时，表示需要新的写请求。如果该参数的请求已处于待决状态，将忽略新请求。
DRIVE	IN	USInt	驱动器地址：该输入是 USS 驱动器的地址。有效范围是驱动器 1 到驱动器 16。
PARAM	IN	UInt	参数编号：此输入指示要写入的驱动器参数。该参数的范围为 0 到 2047。有关如何访问超出该范围的任何参数的详细信息，请参见驱动器手册。
INDEX	IN	UInt	参数索引：该输入指示要写入的驱动器参数索引。索引为一个 16 位值，其中最低有效字节是实际索引值，其范围是 0 到 255。最高有效字节也可被驱动器使用且取决于驱动器。 有关详细信息，请参见驱动器手册。
EEPROM	IN	Bool	存储到驱动器 EEPROM： 该参数为真时，写入驱动器参数的值将存储在驱动器 EEPROM 中。 如果为假，则写操作是临时的，在驱动器循环上电后不会保留。
VALUE	IN	Word,、Int、UInt、DWord,、DInt、UDInt、Real	要写入的参数值。 它必须在 REQ 切换时有效。
USS_DB	IN	Variant	这是对在用户程序中放置 USS_DRV 指令时创建和初始化的背景数据块的引用。
DONE	OUT	Bool	完成： 该参数为真时，表示输入 VALUE 已写入驱动器。 USS_DRV 发现来自驱动器的写响应数据时会设置该位。 满足以下条件之一时复位该位： 通过另一个 USS_WPM 轮询来请求驱动器确认写操作已完成，或者执行接下来两个 USS_DRV 调用的第二个调用时。

参数	参数类型	数据类型	说明
ERROR	OUT	Bool	出现错误： 此参数为真时，表示发生错误，STATUS 输出有效。其它所有输出在出错时均设置为零。 仅在 USS_PORT 指令的 ERROR 和 STATUS 输出中报告通信错误。
STATUS	OUT	UInt	这是请求的状态值。 它表示写请求的结果。 对于有些状态代码，还在“USS_Extended_Error”变量中提供了更多信息。

6.3.1.6 USS 状态代码

在 USS 功能的 STATUS 输出端返回 USS 指令状态代码。

STATUS 值 (W#16#....)	说明
0000	无错误
8180	驱动器响应的长度与从驱动器收到的字符数不匹配。 出错的驱动器编号在“USS_Extended_Error”变量中返回。请参见本表格下方的扩展错误描述。
8181	VALUE 参数不是 Word、Real 或 DWord 数据类型
8182	用户提供了 Word 参数值，但从驱动器响应中收到 DWord 或 Real 值
8183	用户提供了 DWord 或 Real 参数值，但从驱动器响应中收到 Word 值
8184	驱动器响应报文的校验和有错误。 出错的驱动器编号在“USS_Extended_Error”变量中返回。请参见本表格下方的扩展错误描述。
8185	非法的驱动器地址（有效驱动器地址范围：1-16）
8186	速度设定值超出有效范围（有效速度 SP 范围： -200% 到 200%）
8187	对已发送的请求响应了错误的驱动器编号。 出错的驱动器编号在“USS_Extended_Error”变量中返回。请参见本表格下方的扩展错误描述。
8188	指定的 PZD 字长度非法（有效范围 = 2、4、6 或 8 个字）
8189	指定了非法的波特率
818A	参数请求通道正在由该驱动器的另一个请求使用
818B	驱动器尚未对请求和重试做出响应。 出错的驱动器编号在“USS_Extended_Error”变量中返回。请参见本表格下方的扩展错误描述。
818C	驱动器返回有关参数请求操作的扩展错误。 请参见本表格下方的扩展错误描述。

STATUS 值 (W#16#....)	说明
818D	驱动器返回有关参数请求操作的非法访问错误。 有关可能限制参数访问的原因信息，请参见驱动器手册
818E	驱动器尚未初始化： 若从未调用过该驱动器的 USS_DRV，该错误代码将返回到 USS_RPM 或 USS_WPM。 这会防止首次扫描 USS_DRV 的初始化过程覆盖未决的参数读/写请求，因为它会将驱动器初始化为新条目。 要修复该错误，请针对此驱动器编号调用 USS_DRV。
80Ax-80Fx	从由 USS 库调用的 PtP（Point-to-Point，点对点）通信 FB 中返回的特定错误： 这些错误代码值不会被 USS 库修改且在 PtP 指令说明中定义。

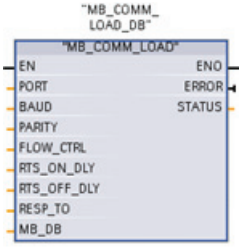
USS 驱动器扩展错误代码

USS 驱动器支持对驱动器的内部参数进行读写访问。 通过该功能可进行驱动器的远程控制和组态。 由于发生类似值超出范围或驱动器当前模式请求非法等错误，驱动器参数访问操作可能会失败。 驱动器会生成在 USS_DRV 背景数据块的“USS_Extended_Error”变量中返回的错误代码值。 该错误代码值仅对 USS_RPM 或 USS_WPM 指令的最后一次执行有效。 当 STATUS 代码值为十六进制的 818C 时，驱动器错误代码将放入“USS_Extended_Error”变量中。 “USS_Extended_Error”的错误代码值取决于驱动器型号。 有关读写参数操作扩展错误代码的描述，请参见驱动器手册。

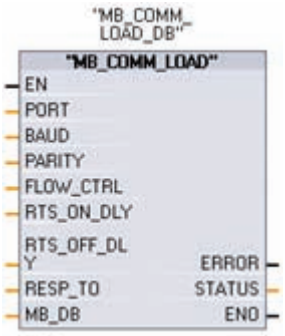
6.3.2 MODBUS

6.3.2.1 MB_COMM_LOAD

LAD



FBD



MB_COMM_LOAD 指令用于组态点对点 (PtP, Point-to-Point) CM 1241 RS485 或 CM 1241 RS232 模块上的端口，以进行 Modbus RTU 协议通信。

参数	参数类型	数据类型	说明
PORT	IN	UInt	通信端口标识符： 在设备配置中安装 CM 模块后，端口标识符会出现在 PORT 功能框连接的助手下拉列表中。 也可以在默认变量表的“常量”(Constants) 选项卡中引用该常量。
BAUD	IN	UDInt	波特率选择： 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200 其它所有值均无效
PARITY	IN	UInt	奇偶校验选择： • 0 – 无 • 1 – 奇校验 • 2 – 偶校验
FLOW_CTRL	IN	UInt	流控制选择： • 0 –（默认）无流控制 • 1 – 硬件流控制，RTS 始终为 ON（不适用于 RS485 端口） • 2 - 带 RTS 切换的硬件流控制
RTS_ON_DLY	IN	UInt	RTS 接通延时选择： • 0 –（默认）从 RTS 激活一直到传送消息的第一个字符之前无延迟 • 1 到 65535 – 从 RTS 激活一直到传送消息的第一个字符之前以毫秒表示的延迟（不适用于 RS-485 端口）。 不管 FLOW_CTRL 选择为何，都将应用 RTS 延迟。
RTS_OFF_DLY	IN	UInt	RTS 关断延时选择： • 0 –（默认）从传送最后一个字符一直到 RTS 转入非活动状态之前无延迟 • 1 到 65535 – 从传送最后一个字符一直到 RTS 转入非活动状态之前以毫秒表示的延迟（不适用于 RS-485 端口）。 不管 FLOW_CTRL 选择为何，都将应用 RTS 延迟。

参数	参数类型	数据类型	说明
RESP_TO	IN	UInt	响应超时： MB_MASTER 允许用于从站响应的时间（以毫秒为单位）。 如果从站在此时间段内未响应，MB_MASTER 将重试请求，或者在发送指定次数的重试请求后终止请求并提示错误。 5 ms 到 65535 ms（默认值 = 1000ms）。
MB_DB	IN	Variant	对 MB_MASTER 或 MB_SLAVE 指令所使用的背景数据块的引用。 在用户程序中放置 MB_SLAVE 或 MB_MASTER 后，DB 标识符会出现在 MB_DB 功能框连接的助手下拉列表中。
ERROR	OUT	Bool	错误： • 0 - 未检测到错误 • 1 – 表示检测到错误并且参数 STATUS 的错误代码有效
STATUS	OUT	Word	端口组态错误代码

可执行 MB_COMM_LOAD 组态端口以使用 Modbus RTU 协议。 组态端口后，即可通过执行 MB_SLAVE 或 MB_MASTER 指令在 Modbus 上通信。

应调用一次 MB_COMM_LOAD 以初始化该端口。 只有某个参数必须更改时，才需要再次调用 MB_COMM_LOAD。 用户可以从启动 OB 调用 MB_COMM_LOAD 并执行它一次，或使用第一个扫描系统标记发起调用以执行它一次。

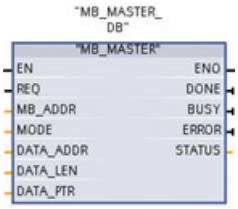
对于每个通信模块中用于 Modbus 通信的每个端口，都必须使用一个 MB_COMM_LOAD 实例来组态。 必须为所用的每个端口都分配一个唯一的 MB_COMM_LOAD 背景数据块。 S7-1200 CPU 被限制为 3 个通信模块。

用户在放置 MB_MASTER 或 MB_SLAVE 指令时分配背景数据块。 指定 MB_COMM_LOAD 指令中的 MB_DB 参数时将引用该背景数据块。

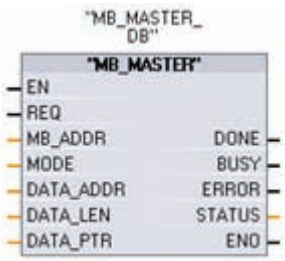
STATUS 值 (W#16#....)	说明
0000	无错误
8180	端口 ID 值无效
8181	波特率值无效
8182	奇偶校验值无效
8183	流控制值无效
8184	响应超时值无效
8185	指向 MB_MASTER 或 MB_SLAVE 的背景数据块的 MB_DB 指针错误

6.3.2.2 MB_MASTER

LAD



FBD



MB_MASTER 指令允许程序作为 Modbus 主站使用点对点 (PtP, Point-to-Point) CM 1241 RS485 或 CM 1241 RS232 模块上的端口进行通信。 可访问一个或多个 Modbus 从站设备中的数据。

用户在程序中放置 MB_MASTER 指令时将分配背景数据块。 指定 MB_MASTER 指令中的 MB_DB 参数时会用到该 MB_SLAVE 背景数据块名称。

参数	参数类型	数据类型	说明
REQ	IN	Bool	请求输入： • 0 – 无请求 • 1 – 请求将数据传送到 Modbus 从站
MB_ADR	IN	USInt	Modbus RTU 站地址： 有效的地址范围： 0 到 247 值 0 被保留用于将消息广播到所有 Modbus 从站。 只有 Modbus 功能代码 05、06、15 和 16 是可用于广播的功能代码。
MODE	IN	USInt	模式选择： 指定请求类型： 读取、写入或诊断 请参见下面的 Modbus 功能表了解详细信息。
DATA_ADDR	IN	UDInt	从站中的起始地址： 指定要在 Modbus 从站中访问的数据的起始地址。 请参见下面的 Modbus 功能表了解有效地址信息。
DATA_LEN	IN	UInt	数据长度： 指定此请求中要访问的位数或字数。 请参见下面的 Modbus 功能表了解有效长度信息。
DATA_PTR	IN	Variant	数据指针： 指向要写入或读取的数据的 CPU DB 地址。 该 DB 必须为“非仅符号访问”DB 类型。 请参见下文的 DATA_PTR 说明。

参数	参数类型	数据类型	说明
NDR	OUT	Bool	新数据就绪： • 0 – 事务未完成 • 1 – 表示 MB_MASTER 指令已完成所请求的有关 Modbus 从站的事务
BUSY	OUT	Bool	忙： • 0 – 无正在进行的 MB_MASTER 事务 • 1 – MB_MASTER 事务正在进行
ERROR	OUT	Bool	错误： • 0 - 未检测到错误 • 1 – 表示检测到错误并且参数 STATUS 提供的错误代码有效
STATUS	OUT	Word	执行条件代码

Modbus 主站通信规则

- 必须先执行 MB_COMM_LOAD 组态端口，然后 MB_MASTER 指令才能与该端口通信。
- 如果要将某个端口用于初始化 Modbus 主站的请求，则 MB_SLAVE 将不能使用该端口。MB_MASTER 执行的一个或多个实例可使用该端口。
- Modbus 指令不使用通信中断事件来控制通信过程。 用户程序必须轮询 MB_MASTER 指令以了解传送和接收的完成情况。
- 如果用户程序操作 Modbus 主站并使用 MB_MASTER 向从站发送请求，则用户必须继续轮询（执行 MB_MASTER）直到返回从站的响应。
- 请从同一个 OB（或 OB 优先等级）调用指定端口的所有 MB_MASTER 执行。

REQ 参数

REQ 值为 FALSE = 无请求

REQ 值为 TRUE = 请求将数据传送到 Modbus 从站。

必须在首次调用 MB_MASTER 执行时通过上升沿触发的触点提供该输入。 沿触发的脉冲将调用该传送请求一次。 在完成由此输入触发的一个请求和响应之前，所有输入将被捕捉并保持不变。

MB_MASTER 将在内部启动状态机，以确保在完成该请求之前不允许其它 MB_MASTER 指令发出请求。

此外，如果在完成请求之前 REQ 输入为 TRUE，从而再次执行 MB_MASTER FB 调用的同一个实例，则不会进行后续传送。 但是，只要请求已完成，因为 REQ 输入设置为 TRUE 而执行 MB_MASTER 时，就会发出新的请求。

DATA_ADDR 和 MODE 参数用于选择 Modbus 功能类型

DATA_ADDR（从站中的起始 Modbus 地址）： 指定要在 Modbus 从站中访问的数据的起始地址。

MB_MASTER 使用 MODE 输入而非功能代码输入。 MODE 和 Modbus 地址范围一起确定实际 Modbus 消息中使用的功能代码。 下表列出了 MBUS_MASTER 参数 MODE、Modbus 功能代码和 Modbus 地址范围之间的对应关系。

MB_MASTER Modbus 功能				
	Modbus 地址参数 DATA_ADDR	地址类型	Modbus 数据长度参数 DATA_LEN	Modbus 功能
模式 0				
读取	00001 到 09999	输出位	1 到 2000	01H
	10001 – 19999	输入位	1 到 2000	02H
	30001 - 39999	输入寄存器	1 到 125	04H
	40001 到 49999 400001 到 465536（扩展）	保持寄存器	1 到 125	03H
模式 1				
写入	00001 到 09999	输出位	1（单个位）	05H
	40001 到 49999 400001 到 465536（扩展）	保持寄存器	1（单个字）	06H
	00001 到 09999	输出位	2 到 1968	15H
	40001 到 49999 400001 到 465536（扩展）	保持寄存器	2 到 123	16H
模式 2				
有些 Modbus 从站不支持使用 Modbus 功能 05H 和 06H 写入单个位或字。 在这些情况下，可通过模式 2 强制使用 Modbus 函数 15H 和 16H 写入单个位和字。				
写入	00001 到 09999	输出位	1 到 1968	15H
	40001 到 49999 400001 到 465536（扩展）	保持寄存器	1 到 123	16H
模式 11 • 从 MB_ADDR 输入引用的 Modbus 从站中读取事件计数器字 • 在 Siemens S7-1200 Modbus 从站中，从站每次从 Modbus 主站收到一个有效的读或写请求（非广播）时，该计数器就会递增。 • 返回的值存储在 DATA_PTR 输入指定的字位置。 • 此模式不需要有效的 DATA_LEN。				

MB_MASTER Modbus 功能
模式 80 • 检查 MB_ADDR 输入引用的 Modbus 从站的通信状态 • MB_MASTER 指令的 NDR 输出位置位时，表示所寻址的 Modbus 从站使用适当的响应数据进行了响应。 • 没有数据返回到用户程序中。 • 此模式不需要有效的 DATA_LEN。
模式 81 • 重置 MB_ADDR 输入引用的 Modbus 从站中的事件计数器（即模式 11 返回的值） • MB_MASTER 指令的 NDR 输出位置位时，表示所寻址的 Modbus 从站使用适当的响应数据进行了响应。 • 没有数据返回到用户程序中。 • 此模式不需要有效的 DATA_LEN。

DATA_PTR 参数

DATA_PTR 参数分别指向读取或写入数据时用到的本地源或目标地址（S7-1200 CPU 中的地址）。 使用 MB_MASTER 指令创建 Modbus 主站时，必须创建全局数据块为读写 Modbus 从站提供数据存储位置。

说明

DATA_PTR 参数必须引用未选中“仅符号访问”(Symbolic access only) 属性框而创建的全局数据块。

在添加新数据块以创建典型的全局 DB 类型时，必须取消选中“仅符号地址”(Symbolic address only) 框。

DATA_PTR 参数的数据块结构

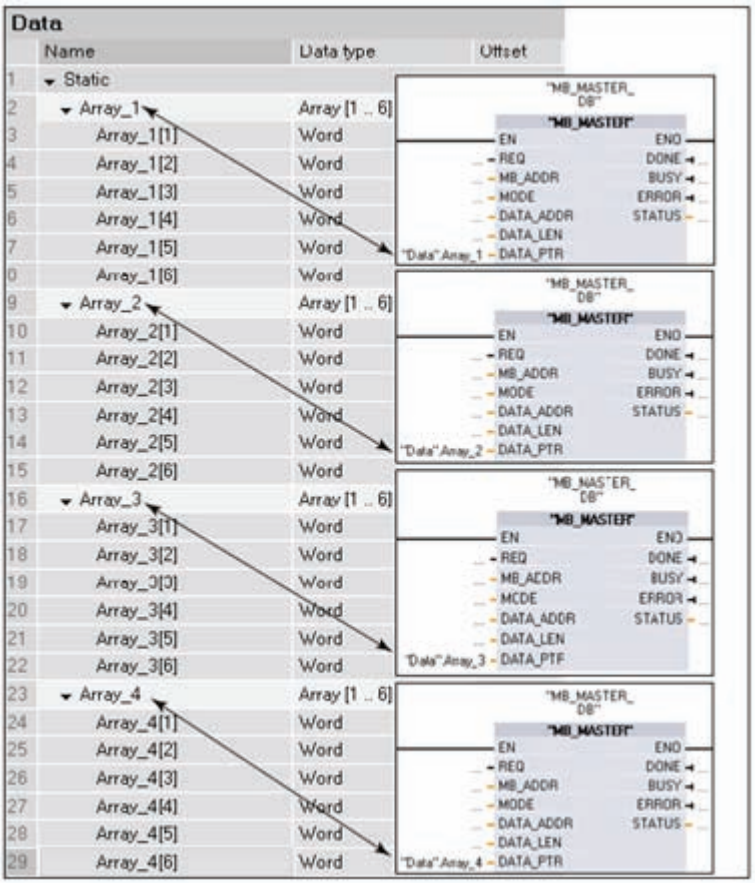
- 这些数据类型对 Modbus 地址 30001 到 39999、40001 到 49999 和 400001 到 465536 的**字读取**有效，对 Modbus 地址 40001 到 49999 和 400001 到 465536 的**字写入**也有效。
  - WORD、UINT 或 INT 数据类型的标准数组，如下所示。
  - 指定的 WORD、UINT 或 INT 结构，其中每个元素都具有唯一的名称和 16 位数据类型。
  - 指定的复杂结构，其中每个元素都具有唯一的名称以及 16 或 32 位数据类型。
- 用于 Modbus 地址 00001 到 09999 和 10001 到 19999 的位读取和写入。
  - 布尔数据类型的标准数组。
  - 具有唯一名称的布尔变量的指定布尔结构。
- 尽管不是必需的，但还是建议每个 MB_MASTER 指令在全局数据块中都具有其自身的单独区域。 此建议的原因在于，如果多个 MB_MASTER 指令读取和写入全局数据块的同一个区域，发生数据损坏的可能性会更大。
- 不要求 DATA_PTR 数据区位于同一个全局数据块中。 可创建一个具有多个区域的数据块供 Modbus 读取、一个数据块供 Modbus 写入或一个数据块用于各个从站。
- 以下实例中的所有数组都以基数为 1 的数组 [1 … ##] 形式创建。 这些数组也可以创建为基数为 0 的数组 [0 … ###] 或基数为 0 和基数为 1 的混合数组。

访问 DATA_PTR 全局数据块的 MB_MASTER 指令实例

下面显示的全局数据块示例使用了 4 个名称唯一的数组（每个数组包含 6 个字元素）来存储 Modbus 请求数据。 尽管此实例中的数据数组大小相同，但数组可以是任意大小，显示为相同大小是为了简化实例。 也可以用包括更多描述性变量名称和混合数据类型的数据结构来代替其中的每个数组。 MB_SLAVE 指令 (页 232)的 HR_DB 参数描述中提供了备选数据结构的实例。

下面的 MB_MASTER 指令实例仅说明 DATA_PTR 参数，未说明其它必要的参数。 该实例的目的说明 MB_MASTER 指令将如何使用 DATA_PTR 数据块。

箭头指示各个数组如何与不同的 MB_MASTER 指令相关联。



任何数组或结构的第一个元素始终为某个 Modbus 读取或写入活动的第一个源或目标。下面的所有情形都是基于上图的。

情形 1： 如果第一个 MB_MASTER 指令从任意有效的 Modbus 从站的 Modbus 地址 40001 中读取了 3 个字的数据，则会发生以下情况。

地址 40001 的字存储在 "Data".Array_1[1] 中。

地址 40002 的字存储在 "Data".Array_1[2] 中。

地址 40003 的字存储在 "Data".Array_1[3] 中。

情形 2： 如果第一个 MB_MASTER 指令从任意有效的 Modbus 从站的 Modbus 地址 40015 中读取了 4 个字的数据，则会发生以下情况。

地址 40015 的字存储在 "Data".Array_1[1] 中。

地址 40016 的字存储在 "Data".Array_1[2] 中。

地址 40017 的字存储在 "Data".Array_1[3] 中。

地址 40018 的字存储在 "Data".Array_1[4] 中。

情形 3： 如果第二个 MB_MASTER 指令从任意有效的 Modbus 从站的 Modbus 地址 30033 中读取了 2 个字的数据，则会发生以下情况。

地址 30033 的字存储在 "Data".Array_2[1] 中。

地址 30034 的字存储在 "Data".Array_2[2] 中。

情形 4： 如果第三个 MB_MASTER 指令将 4 个字的数据写入任意 Modbus 从站的 Modbus 地址 40050 中，则会发生以下情况。

"Data".Array_3[1] 中的字写入 Modbus 地址 40050。

"Data".Array_3[2] 中的字写入 Modbus 地址 40051。

"Data".Array_3[3] 中的字写入 Modbus 地址 40052。

"Data".Array_3[4] 中的字写入 Modbus 地址 40053。

情形 5： 如果第三个 MB_MASTER 指令将 3 个字的数据写入任意 Modbus 从站的 Modbus 地址 40001 中，则会发生以下情况。

"Data".Array_3[1] 中的字写入 Modbus 地址 40001。

"Data".Array_3[2] 中的字写入 Modbus 地址 40002。

"Data".Array_3[3] 中的字写入 Modbus 地址 40003。

情形 6： 如果第四个 MB_MASTER 指令使用模式 11 从任意有效的 Modbus 从站检索有效消息计数，则会发生以下情况。

计数字存储在 "Data".Array_4[1] 中。

使用 DATA_PTR 输入的字位置进行的位读取和写入实例

表格 6- 1　情形 7：从 Modbus 地址 00001 开始读取 4 个输出位

MB_MASTER 输入值			Modbus 从站值	
MB_ADDR	27（从站实例）		**00001**	**ON**
MODE	0（读）		**00002**	**ON**
DATA_ADDR	00001（输出）		**00003**	**OFF**
DATA_LEN	4		**00004**	**ON**
DATA_PTR	"Data".Array_4		00005	ON
			00006	OFF
			00007	ON
			00008	OFF

Modbus 请求之后的 "Data".Array_4[1] 值	
MS（最高有效）字节	LS（最低有效）字节
xxxx-1011	xxxx-xxxx
x 表示数据没有变化	

表格 6- 2　情形 8：从 Modbus 地址 00003 开始读取 12 个输出位

MB_MASTER 输入值			Modbus 从站值				
MB_ADDR	27（从站实例）		00001	ON		**00010**	**ON**
MODE	0（读）		00002	ON		**00011**	**OFF**
DATA_ADDR	00003（输出）		**00003**	**OFF**		**00012**	**OFF**
DATA_LEN	12		**00004**	**ON**		**00013**	**ON**
DATA_PTR	"Data".Array_4		**00005**	**ON**		**00014**	**OFF**
			00006	**OFF**		00015	ON
			00007	**ON**		00016	ON
			00008	**ON**		00017	OFF
			00009	**OFF**		00018	ON

Modbus 请求之后的 "Data".Array_4[1] 值	
MS 字节	LS 字节
1011-0110	xxxx-0100-
x 表示数据没有变化	

表格 6- 3 情形 9： 从 Modbus 地址 00001 开始写入 5 个输出位

MB_MASTER 输入值			之前的从站输出			之后的从站输出
MB_ADDR	27（从站实例）		**00001**	**ON**		**OFF**
MODE	1（写）		**00002**	**ON**		ON
DATA_ADDR	00001（输出）		**00003**	**OFF**		ON
DATA_LEN	5		**00004**	**ON**		**OFF**
DATA_PTR	"Data".Array_4		**00005**	**ON**		ON
			00006	OFF		不变
			00007	ON		不变
			00008	ON		不变
			00009	OFF		不变

Modbus 写请求的 "Data".Array_4[1] 值	
MS 字节	LS 字节
xxx1-0110	xxxxx-xxxx
x 表示数据未在 Modbus 请求中使用	

表格 6- 4　情形 10：从 Modbus 地址 00003 开始读取 22 个输出位

MB_MASTER 输入值			Modbus 从站值				
MB_ADDR	27（从站实例）		00001	ON		**00014**	**ON**
MODE	0（读）		00002	ON		**00015**	**OFF**
DATA_ADDR	00003（输出）		**00003**	**OFF**		**00016**	**ON**
DATA_LEN	22		**00004**	**ON**		**00017**	**ON**
DATA_PTR	"Data".Array_4		**00005**	**ON**		**00018**	**OFF**
			00006	**OFF**		**00019**	**ON**
			00007	**ON**		**00020**	**ON**
			00008	**ON**		**00021**	**OFF**
			00009	**ON**		**00022**	**ON**
			00010	**OFF**		**00023**	**ON**
			00011	**OFF**		**00024**	**OFF**
			00012	**ON**		00025	OFF
			00013	**OFF**		00026	ON

Modbus 请求之后的 "Data".Array_4[1] 值	
MS 字节	LS 字节
0111-0110	0110-1010

Modbus 请求之后的 "Data".Array_4[2] 值	
MS 字节	LS 字节
xx01-1011	xxxx-xxxx
x 表示数据没有变化	

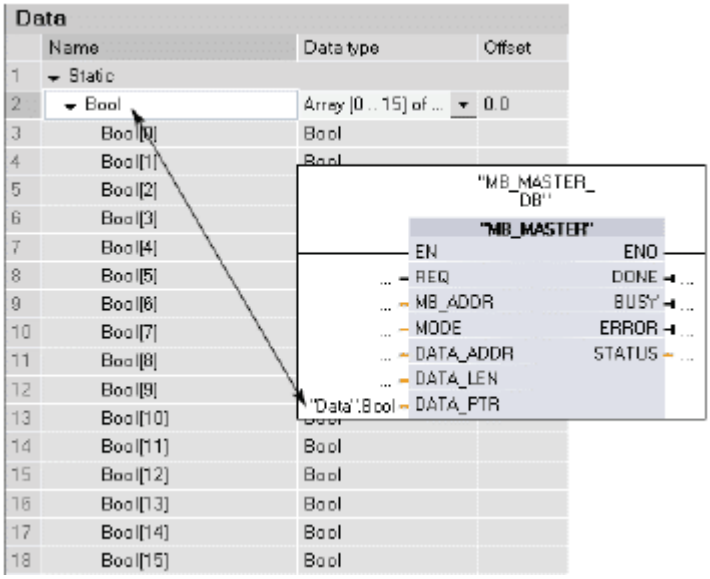
使用 DATA_PTR 输入的 BOOL 位置进行的位读取和写入实例

尽管 Modbus 对位地址位置的读取和写入可通过使用字位置进行处理，但也可将 DATA_PTR 区域组态为布尔数据类型、结构或数组，为通过 MB_MASTER 指令读取或写入的第一个位提供直接的一对一关系。

如果使用布尔数组或结构，建议用户将数据大小设置为 8 位（基于字节）的倍数。 例如，如果创建一个 10 位的布尔数组，STEP 7 Basic 软件将在全局数据块中为这 10 位分配 16 个位（2 个字节）。 在相应的数据块内部，这些位将存储成“字节 1 [xxxx xxxx] 字节 2 [---- --xx]”，其中 x 表示可访问的数据位置，“–”表示不可访问的位置。 最多允许 16 位长度的 Modbus 请求，但更高的 6 位将放置在字节 2 存储位置中，无法被用户程序引用和访问。

布尔区域可创建成布尔值的数组或布尔变量的结构。 两种方法作用方式相同，仅在用户程序中创建和访问的方式不同。

下面的全局数据块编辑器视图显示了以 0 为基数的 16 个布尔值的单个数组。该数组也可创建为基数为 1 的数组。 箭头说明该数组如何与 MB_MASTER 指令相关联。



情形 11 和 12 说明 Modbus 地址与布尔数组地址的对应关系。

表格 6- 5 情形 11： 从 Modbus 地址 00001 开始写入 5 个输出位

MB_MASTER 输入值			之前的从站输出		DATA_PTR 数据	之后的从站输出
MB_ADDR	27（从站实例）		00001	ON	"Data".Bool[0]=FALSE	OFF
MODE	1（写）		00002	ON	"Data".Bool[1]=TRUE	ON
DATA_ADDR	00001（输出）		00003	OFF	"Data".Bool[2]=TRUE	ON
DATA_LEN	5		00004	ON	"Data".Bool[3]=FALSE	OFF
DATA_PTR	"Data".Bool		00005	ON	"Data".Bool[4]=FALSE	OFF
			00006	OFF		不变
			00007	ON		不变
			00008	OFF		不变

表格 6- 6 情形 12： 从 Modbus 地址 00004 开始读取 15 个输出位

MB_MASTER 输入值			Modbus 从站值		之后的 DATA_PTR 数据
MB_ADDR	27（从站实例）		00001	ON	
MODE	0（读）		00002	ON	
DATA_ADDR	00003（输出）		**00003**	**OFF**	"Data".Bool[0]=FALSE
DATA_LEN	15		**00004**	**ON**	"Data".Bool[1]=TRUE
DATA_PTR	"Data".Bool		**00005**	**ON**	"Data".Bool[2]=TRUE
			00006	**OFF**	"Data".Bool[3]=FALSE
			00007	**ON**	"Data".Bool[4]=TRUE
			00008	**ON**	"Data".Bool[5]=TRUE
			00009	**ON**	"Data".Bool[6]=TRUE
			00010	**OFF**	"Data".Bool[7]=FALSE
			00011	**OFF**	"Data".Bool[8]=FALSE
			00012	**ON**	"Data".Bool[9]=TRUE
			00013	**OFF**	"Data".Bool[10]=FALSE
			00014	**ON**	"Data".Bool[11]=TRUE
			00015	**OFF**	"Data".Bool[12]=FALSE
			00016	**ON**	"Data".Bool[13]=TRUE
			00017	**ON**	"Data".Bool[14]=TRUE
			00018	OFF	
			00019	ON	

条件代码

STATUS 值 (W#16#....)	说明
0000	无错误
80C8	指定的响应超时时间（指 RCVTIME 或 MSGTIME）为 0。
80D1	接收方发出了暂停主动传输的流控制请求并且在指定的等待时间内未重新激活该传输。 在硬件流控制期间，如果接收方在指定的等待时间内没有声明 CTS，也会产生该错误。
80D2	传送请求中止，因为没有从 DCE 收到任何 DSR 信号。
80E0	因接收缓冲区已满，消息被终止。
80E1	因出现奇偶校验错误，消息被终止。
80E2	因组帧错误，消息被终止。
80E3	因出现超限错误，消息被终止。
80E4	因指定长度超出总缓冲区大小，消息被终止。
8180	端口 ID 值无效
8186	Modbus 站地址无效
8188	模式值无效或只读从站地址区的写模式无效
8189	数据地址值无效
818A	数据长度值无效
818B	• 指向本地数据源/目标的指针无效： 大小不正确
818C	指向安全 DB 类型的 DATA_PTR（必须为典型 DB 类型）的指针
8200	端口正忙于处理传送请求

6.3.2.3 MB_SLAVE

MB_SLAVE 指令允许程序作为 Modbus 从站使用点对点 (PtP, Point-to-Point) CM 1241 RS485 或 CM 1241 RS232 模块上的端口进行通信。 Modbus RTU 主站可以发出请求，然后程序通过执行 MB_SLAVE 来响应。

在程序中放置 MB_SLAVE 指令时，必须分配唯一的背景数据块。 指定 MB_COMM_LOAD 指令中的 MB_DB 参数时会用到该 MB_SLAVE 背景数据块名称。

Modbus 通信功能代码（1、2、4、5 和 15）可以在 PLC 输入过程映像及输出过程映像中直接读写位和字。 下表给出了 Modbus 地址与 CPU 中的过程映像的映射关系。

MB_SLAVE Modbus 功能						S7-1200	
代码	功能	数据区	地址范围			数据区	CPU 地址
01	读位	输出	1	到	8192	输出过程映像	Q0.0 到 Q1023.7
02	读位	输入	10001	到	18192	输入过程映像	I0.0 到 I1023.7
04	读字	输入	30001	到	30512	输入过程映像	IW0 到 IW1022
05	写位	输出	1	到	8192	输出过程映像	Q0.0 到 Q1023.7
15	写位	输出	1	到	8192	输出过程映像	Q0.0 到 Q1023.7

Modbus 通信功能代码（3、6、16）使用单独且唯一的 Modbus 保持寄存器数据块，必须先创建该数据块，然后才能指定 MB_SLAVE 指令的 MB_HOLD_REG 参数。 下表给出了 Modbus 保持寄存器与 PLC 中的 MB_HOLD_REG DB 地址的映射关系。

MB_SLAVE Modbus 功能				S7-1200	
代码	功能	数据区	地址范围	CPU DB 数据区	CPU DB 地址
03	读字	保持寄存器	40001 到 49999	MB_HOLD_REG	字 1 到 9999
			400001 到 465535		字 1 到 65535
06	写字	保持寄存器	40001 到 49999	MB_HOLD_REG	字 1 到 9999
			400001 到 465535		字 1 到 65535
16	写字	保持寄存器	40001 到 49999	MB_HOLD_REG	字 1 到 9999
			400001 到 465535		字 1 到 65535

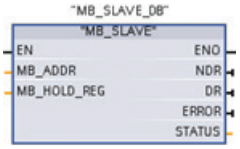
下表说明了支持的 Modbus 诊断功能。

S7-1200 MB_SLAVE Modbus 诊断功能		
代码	子功能	说明
08	0000H	返回查询数据回送测试： MB_SLAVE 将向 Modbus 主站回送接收到的数据字。
08	000AH	清除通信事件计数器： MB_SLAVE 将清除用于 Modbus 功能 11 的通信事件计数器。
11		获取通信事件计数器： MB_SLAVE 使用内部通信事件计数器来记录发送到 Modbus 从站的 Modbus 成功读取和写入请求次数。 该计数器不会因功能 8、功能 11 或广播请求而增加。 同样也不会因任何导致通信错误（例如，奇偶校验错误或 CRC 错误）的请求而增加。

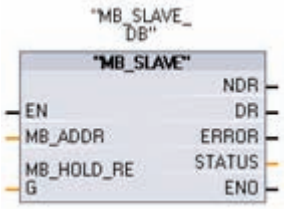
MB_SLAVE 支持来自任何 Modbus 主站的广播写入请求，只要该请求是用于访问有效位置的请求即可。

不管请求是否有效，MB_SLAVE 都不对 Modbus 主站的广播请求做出任何响应。

LAD



FBD



参数	参数类型	数据类型	说明
MB_ADDR	IN	USINT	Modbus RTU 地址（1 到 247）： Modbus 从站的站地址。
MB_HOLD_REG	IN	VARIANT	指向 Modbus 保持寄存器 DB 的指针。 保持寄存器 DB 必须为典型的全局 DB。请参见下文的 MB_HOLD_REG 说明。
NDR	OUT	BOOL	新数据就绪： • 0 – 无新数据 • 1 – 表示 Modbus 主站已写入新数据

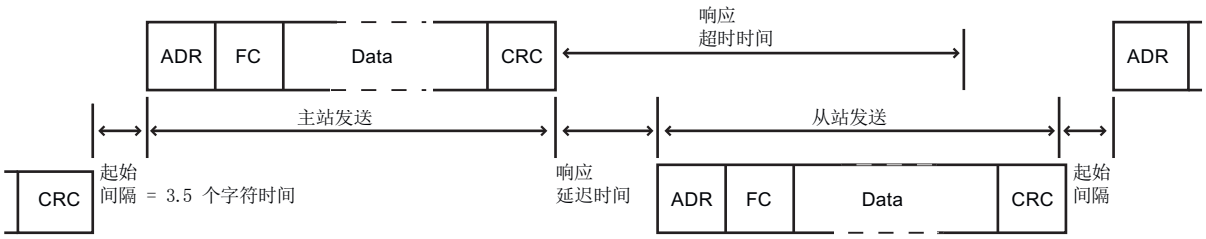
参数	参数类型	数据类型	说明
DR	OUT	BOOL	数据读取： • 0 – 无数据读取 • 1 – 表示 Modbus 主站已读取数据
ERROR	OUT	BOOL	错误： • 0 - 未检测到错误 • 1 – 表示检测到错误并且参数 STATUS 提供的错误代码有效。
STATUS	OUT	WORD	错误代码

Modbus 从站通信规则

- 必须先执行 MB_COMM_LOAD 组态端口，然后 MB_SLAVE 指令才能与该端口通信。
- 如果某个端口作为从站响应 Modbus 主站，则 MB_MASTER 无法使用该端口。 对于给定端口，只能使用一个 MB_SLAVE 执行实例。
- Modbus 指令不使用通信中断事件来控制通信过程。 用户程序必须通过轮询 MB_SLAVE 指令以了解传送和接收的完成情况来控制通信过程。
- MB_SLAVE 必须以一定的速率定期执行，以便能够及时响应来自 Modbus 主站的进入请求。
- 每次扫描都应从程序循环 OB 中调用 MB_SLAVE。

操作

必须周期性地执行 MB_SLAVE，才能接收来自 Modbus 主站的每个请求并随之按要求响应。 MB_SLAVE 的执行频率取决于 Modbus 主站的响应超时时间。 下图对此进行了说明。



响应超时时间是 Modbus 主站等待 Modbus 从站开始响应的时间。 该时间段不是由 Modbus 协议定义的，而是属于每个 Modbus 主站的一个参数。 必须基于用户 Modbus 主站的具体参数确定 MB_SLAVE 的执行频率（相邻两次执行之间的时间）。 在 Modbus 主站的响应超时时间内至少应执行两次 MB_SLAVE。

MB_HOLD_REG 参数实例

MB_HOLD_REG 是指向 Modbus 保持寄存器数据块的指针。 该 DB 用于保存允许 Modbus 主站访问（读或写）的数据值。 在将其用于 MB_SLAVE 指令之前，必须先创建该数据块并分配用于读写操作的数据类型结构。

说明

Modbus 保持寄存器数据块必须引用未选中“仅符号访问”(Symbolic access only) 属性框而创建的全局数据块。

在添加新数据块以创建典型的全局 DB 类型时，必须取消选中“仅符号地址”(Symbolic address only) 框。

保持寄存器可使用以下 DB 数据结构：

- 标准的字数组
- 指定的字结构
- 指定的复杂结构

下面的程序实例介绍了如何使用 MB_HOLD_REG 参数处理这些 DB 数据结构。

实例 1 - 标准的字数组

该保持寄存器实例是一个字数组。 数据类型分配可以更改为其它字大小类型（INT 和 UINT）。

优点：

- 可以非常快速又简单地创建这种类型的保持寄存器结构。
- 访问数据元素的程序逻辑经过简化。

缺点：

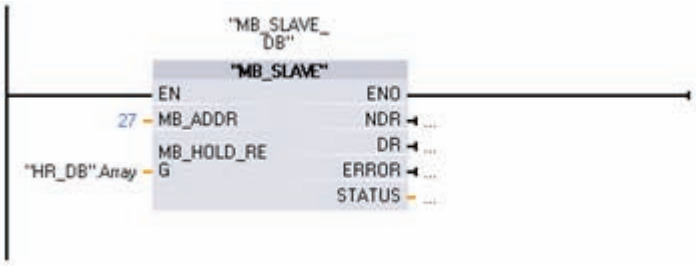
- 尽管可以通过程序用符号名称（"HR_DB"."Array"[1] 到 "HR_DB"."Array"[10]）来引用各数组元素，但这些名称不能描述数据的内部功能。
- 该数组只能由一种数据类型组成。 在用户程序中可能需要通过严格类型控制来进行类型转换。

以下是字数组结构在数据块编辑器中的显示方式。

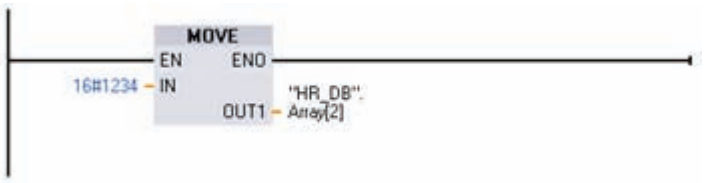
HR_DB

	Name	Data type	Offset	Default value	Initial value	Retain	Comment
1	Static						
2	Array	Array [1..10] of W..	0.0				40001 to 40010
3	Array[1]	Word		W#16#0000	W#16#0000		
4	Array[2]	Word		W#16#0000	W#16#0000		
5	Array[3]	Word		W#16#0000	W#16#0000		
6	Array[4]	Word		W#16#0000	W#16#0000		
7	Array[5]	Word		W#16#0000	W#16#0000		
8	Array[6]	Word		W#16#0000	W#16#0000		
9	Array[7]	Word		W#16#0000	W#16#0000		
10	Array[8]	Word		W#16#0000	W#16#0000		
11	Array[9]	Word		W#16#0000	W#16#0000		
12	Array[10]	Word		W#16#0000	W#16#0000		

下图显示了数组是如何分配给 MB_SLAVE 指令的 MB_HOLD_REG 输入的。



数组的每个元素都可通过符号名来访问，如下所示。 本例中，新值被移动到数组内对应 Modbus 地址 40002 的第二个元素中。



数组中的每个字（在数据块中定义）都为 MB_SLAVE 指令提供 Modbus 保持寄存器地址。 本例中，由于数组中只有 10 个元素，因此只有 10 个 Modbus 寄存器地址可用于 MB_SLAVE 指令并可被 Modbus 主站访问。

数组元素名称与 Modbus 地址间的关系如下所示。

"HR_DB".Array[1]	Modbus 地址 40001
" HR_DB ". Array[2]	Modbus 地址 40002
" HR_DB ". Array[3]	Modbus 地址 40003
...	...
" HR_DB ". Array[9]	Modbus 地址 40009
" HR_DB ". Array[10]	Modbus 地址 40010

实例 2 - 指定的字结构

该保持寄存器实例是具有描述性符号名的一系列字。

优点：

- 每个结构元素都有一个描述性名称，并被分配了一个特定的数据类型。

缺点：

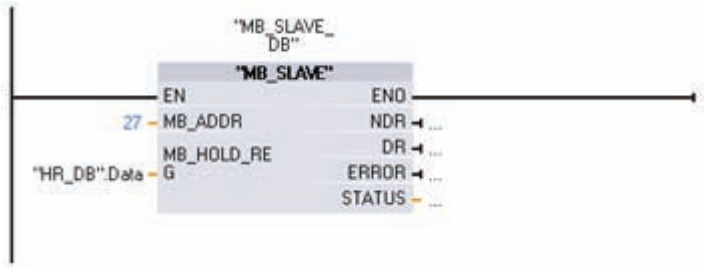
- 创建这种类型的结构要比创建标准字数组花费更长时间。
- 在用户程序中使用元素时需要额外的符号引用。 其中，以 "HR_DB".Array[0] 方式引用简单数组的第一个元素，而需要以 "HR_DB".Data.Temp_1 方式引用这种类型的第一个元素。

以下是指定的字结构在数据块编辑器中的显示方式。 每个元素都有唯一的名称且可以是 WORD、UINT 或 INT。

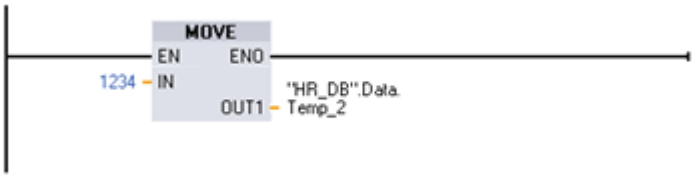
HR_DB

	Name	Data type	Offset	Initial value	Retain	Comment
1	Static					
2	Data	Struct	0.0			
3	Temp_1	Int	0.0	0		40001
4	Temp_2	Int	2.0	0		40002
5	Temp_3	Int	4.0	0		40003
6	Good_Count	UInt	6.0	0		40004
7	Bad_Count	UInt	8.0	0		40005
8	Rework_Count	UInt	10.0	0		40006
9	Line_Stops	UInt	12.0	0		40007
10	Avg_Time	UInt	14.0	0		40008
11	Code_1	Word	16.0	0		40009
12	Code_2	Word	18.0	0		40010

下图显示了上述数据结构在程序中是如何分配给 MB_SLAVE 指令的 MB_HOLD_REG 输入的。



数组的每个元素都可通过其符号名来访问，如下所示。 本例中，新值被移动到数组内对应 Modbus 地址 40002 的第二个元素中。



数据元素名称与 Modbus 地址间的关系如下所示。

"HR_DB".Data.Temp_1	Modbus 地址 40001
"HR_DB".Data.Temp_2	Modbus 地址 40002
"HR_DB".Data.Temp_3	Modbus 地址 40003
"HR_DB".Data.Good_Count	Modbus 地址 40004
"HR_DB".Data.Bad_Count	Modbus 地址 40005
"HR_DB".Data.Rework_Count	Modbus 地址 40006
"HR_DB".Data.Line_Stops	Modbus 地址 40007
"HR_DB".Data.Avg_Time	Modbus 地址 40008
"HR_DB".Data.Code_1	Modbus 地址 40009
"HR_DB".Data.Code_2	Modbus 地址 40010

实例 3 - 指定的复杂结构

该保持寄存器实例是具有描述性符号名的一系列混合数据类型。

优点：

- 每个结构元素都有一个描述性名称，并被分配了一个特定的数据类型。
- 可用来直接传送不是基于字的数据类型。

缺点：

- 创建这种类型的结构要比创建标准字数组花费更长时间。
- 需组态 Modbus 主站以接受从 Modbus 从站收到的数据。 如下图所示，Temp_1 是一个 4 字节的实数值。 接收主站需要能够将接收到的 2 个字重新组合成所需的实数值。
- 在用户程序中使用元素需要额外的符号引用。 其中，以 "HR_DB".Array[0] 方式引用简单数组的第一个元素，而需要以 "HR_DB".Data.Temp_1 方式引用这种类型的第一个元素。

以下是指定的复杂结构在数据块编辑器中的显示方式。 每个元素都有唯一的名称且其大小和数据类型可以不相同。

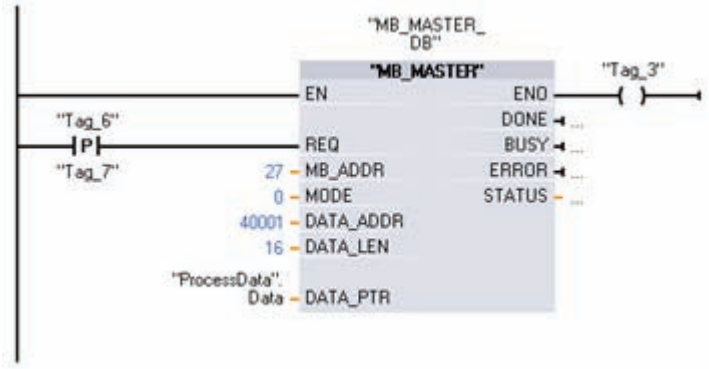
HR_DB

	Name	Data type	Offset	Initial value	Retain	Comment
1	Static					
2	Data	Struct	0.0			Modbus addresses 40001 to 400016
3	Temp_1	Real	0.0	0.0		40001 and 40002
4	Temp_2	Real	4.0	0.0		40003 and 40004
5	Good_Count	UDInt	8.0	0		40005 and 40006
6	Bad_Count	UDInt	12.0	0		40007 and 40008
7	Rework_Count	UDInt	16.0	0		40009 and 40010
8	Line_Stops	Int	20.0	0		40011
9	Avg_Time	Int	22.0	0		40012
10	Long_Code	DWord	24.0	0		40013 and 40014
11	Code_1	Word	28.0	0		40015
12	Code_2	Word	30.0	0		40016

数据元素名称与 Modbus 地址间的关系如下所示。

"HR_DB".Data.Temp_1	Modbus 地址 40001 和 40002
"HR_DB".Data.Temp_2	Modbus 地址 40003 和 40004
"HR_DB".Data.Good_Count	Modbus 地址 40005 和 40006
"HR_DB".Data.Bad_Count	Modbus 地址 40007 和 40008
"HR_DB".Data.Rework_Count	Modbus 地址 40009 和 40010
"HR_DB".Data.Line_Stops	Modbus 地址 400011
"HR_DB".Data.Avg_Time	Modbus 地址 400012
"HR_DB".Data.Long_Code	Modbus 地址 40013 和 40014
"HR_DB".Data.Code_1	Modbus 地址 40015
"HR_DB".Data.Code_2	Modbus 地址 40016

另一个用作 Modbus 主站的 S7-1200 CPU 可以使用 MB_Master 指令和相同的数据结构接收来自用作 Modbus 从站的 S7-1200 CPU 的数据块。 该 Modbus 主站指令会将全部 16 个字的数据直接从从站的 HR_DB 数据块复制到主站的 ProcessData 数据块，如下图所示。



ProcessData

	Name	Data type	Offset	Initial value	Retain	Comment
1	Static					
2	Data	Struct	0.0			Modbus addresses 40001 to 400016
3	Temp_1	Real	0.0	0.0		40001 and 40002
4	Temp_2	Real	4.0	0.0		40003 and 40004
5	Good_Count	UDInt	8.0	0		40005 and 40006
6	Bad_Count	UDInt	12.0	0		40007 and 40008
7	Rework_Count	UDInt	16.0	0		40009 and 40010
8	Line_Stops	Int	20.0	0		40011
9	Avg_Time	Int	22.0	0		40012
10	Long_Code	DWord	24.0	0		40013 and 40014
11	Code_1	Word	28.0	0		40015
12	Code_2	Word	30.0	0		40016

可使用一系列的 Modbus 主站 Data_PTR 数据块位置传送来自多个 Modbus 从站的相同或不同的结构。

条件代码

STATUS 值 (W#16#....)	说明
80C8	指定的响应超时时间（指 RCVTIME 或 MSGTIME）为 0
80D1	接收方发出了暂停主动传输的流控制请求并且在指定的等待时间内未重新激活该传输。 在硬件流控制期间，如果接收方在指定的等待时间内没有声明 CTS，也会产生该错误。
80D2	传送请求中止，因为没有从 DCE 收到任何 DSR 信号
80E0	因接收缓冲区已满，消息被终止
80E1	因出现奇偶校验错误，消息被终止
80E2	因组帧错误，消息被终止
80E3	因出现超限错误，消息被终止
80E4	因指定长度超出总缓冲区大小，消息被终止

STATUS 值 (W#16#....)	说明	
8180	端口 ID 值无效	
8186	Modbus 站地址无效	
8187	指向 MB_HOLD_REG DB 的指针无效	
818C	指向安全 DB 类型的 MB_HOLD_REG DB（必须为典型 DB 类型）的指针	
	发送到 Modbus 主站的响应代码 (B#16#..)	
8380	无响应	CRC 错误
8381	01	不支持此功能代码
8382	无响应	数据长度错误
8383	02	数据地址错误
8384	03	数据值错误
8385	03	不支持此数据诊断代码值（功能代码 08）

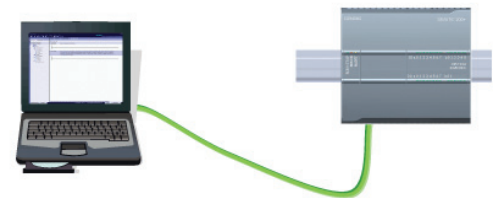
7 PROFINET

S7-1200 CPU 具有一个集成的 PROFINET 端口，支持以太网和基于 TCP/IP 的通信标准。 S7-1200 CPU 支持以下应用协议：

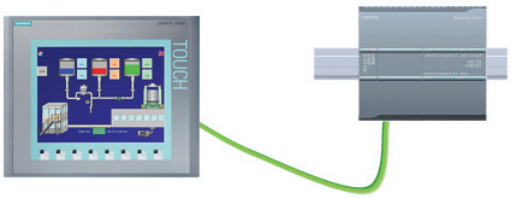
- 传输控制协议 (TCP)
- ISO on TCP (RFC 1006)

S7-1200 CPU 可以使用 TCP 通信协议与其它 S7-1200 CPU、STEP 7 Basic 编程设备、HMI 设备和非 Siemens 设备通信。 有两种使用 PROFINET 通信的方法：

- 直接连接： 在使用连接到单个 CPU 的编程设备、HMI 或另一个 CPU 时采用直接通信。
- 网络连接： 在连接两个以上的设备（例如，CPU、HMI、编程设备和非西门子设备）时采用网络通信。



直接连接： 编程设备连接到 S7-1200 CPU



直接连接： HMI 连接到 S7-1200 CPU

直接连接： 一个 S7-1200 CPU 连接到另一个 S7-1200 CPU

网络连接： 两个以上的设备通过 CSM1277 以太网交换机 ① 连接在一起

编程设备或 HMI 与 CPU 之间的直接连接不需要以太网交换机。 含有两个以上的 CPU 或 HMI 设备的网络需要以太网交换机。 安装在机架上的 Siemens CSM1277 4 端口以太网交换机可用于连接 CPU 和 HMI 设备。 S7-1200 CPU 上的 PROFINET 端口不包含以太网交换设备。

PROFINET 端口的最大连接数

CPU 上的 PROFINET 端口支持以下并发通信连接。

- 3 个用于 HMI 与 CPU 通信的连接
- 1 个用于编程设备 (PG) 与 CPU 通信的连接
- 8 个使用传输块 (T-block) 指令（TSEND_C、TRCV_C、TCON、TDISCON、TSEN、TRCV）实现 S7-1200 程序通信的连接
- 3 个用于被动 S7-1200 CPU 与主动 S7 CPU 通信的连接
  - 主动 S7 CPU 使用 GET 和 PUT 指令（S7-300 和 S7-400）或 ETHx_XFER 指令 (S7-200)。
  - 主动 S7-1200 通信连接只能使用传输块 (T-block) 指令。

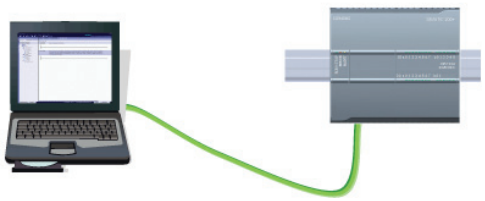
被动 ISO 和 TCP 通信的 TSAP 或端口号限制

如果使用“TCON”指令设置并建立被动通信连接，则下列端口地址将受到限制，不应该使用：

- ISO TSAP（被动）：01.00、01.01、02.00、02.01、03.00、03.01
- TCP 端口（被动）：5001、102、123、20、21、25、34962、34963、34964、80

7.1 与编程设备通信

CPU 可以与网络上的 STEP 7 Basic 编程设备进行通信。



在 CPU 和编程设备之间建立通信时请考虑以下几点：

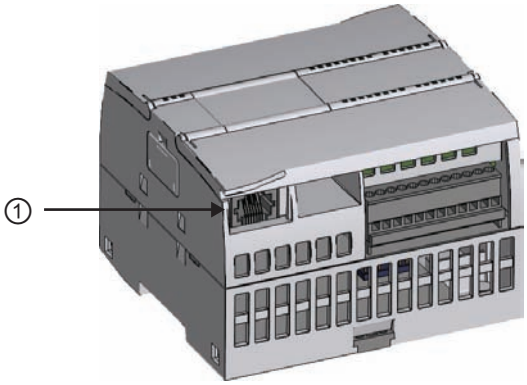
- 组态/设置： 需要进行硬件配置。
- 一对一通信不需要以太网交换机；网络中有两个以上的设备时需要以太网交换机。

7.1.1 建立硬件通信连接

PROFINET 接口可在编程设备和 CPU 之间建立物理连接。 由于 CPU 内置了自动跨接功能，所以对该接口既可以使用标准以太网电缆，又可以使用跨接以太网电缆。 将编程设备直接连接到 CPU 时不需要以太网交换机。

要在编程设备和 CPU 之间创建硬件连接，请按以下步骤操作：

1. 安装 CPU (页 28)。
2. 将以太网电缆插入下图所示的 PROFINET 端口中。
3. 将以太网电缆连接到编程设备上。



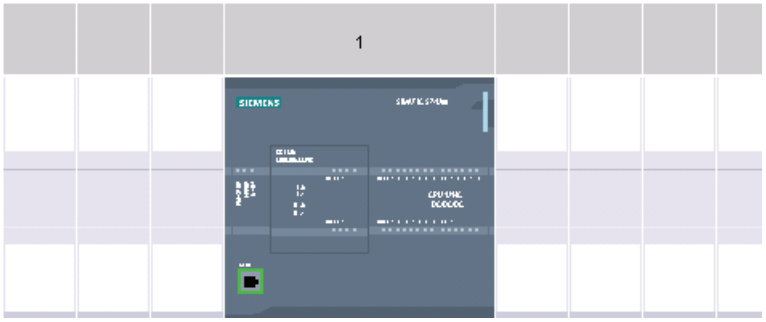
① PROFINET 端口

可选配张力消除装置以加固 PROFINET 连接。

7.1.2 配置设备

如果已经创建带有 CPU 的项目，请在 TIA 门户中打开该项目。

如果没有，请创建项目并在机架中插入 CPU (页 76)。 在下面的项目中，在 TIA 门户的“设备视图”(Device View) 中显示了一个 CPU。



7.1.3 分配 Internet 协议 (IP) 地址

7.1.3.1 为编程设备和网络设备分配 IP 地址

如果编程设备使用板载适配器卡连接到工厂 LAN（可能是万维网），则 CPU 与编程设备板载适配器卡的 IP 地址网络 ID 和子网掩码必须完全相同。网络 ID 是 IP 地址的第一部分（前三个八位位组）（例如，**211.154.184**.16），它决定用户所在的 IP 网络。 子网掩码的值通常为 **255.255.255.0**；然而由于您的计算机处于工厂 LAN 中，子网掩码可能有不同的值（例如，**255.255.254.0**）以设置唯一的子网。子网掩码通过与设备 IP 地址进行数学 AND 运算来确定 IP 子网的边界。

说明

在万维网环境下，编程设备、网络设备和 IP 路由器可与全世界通信，但必须分配唯一的 IP 地址以避免与其它网络用户冲突。 请联系公司 IT 部门熟悉工厂网络的人员分配 IP 地址。

如果编程设备使用连接到独立网络的以太网转 USB 适配器卡，则 CPU 与编程设备的以太网转 USB 适配器卡的 IP 地址网络 ID 和子网掩码必须完全相同。网络 ID 是 IP 地址的第一部分（前三个八位位组）（例如，**211.154.184**.16），它决定用户所在的 IP 网络。子网掩码的值通常为 **255.255.255.0**。子网掩码通过与设备 IP 地址进行数学 AND 运算来确定 IP 子网的边界。

说明

当不想将 CPU 连入公司 LAN 时，非常适合使用以太网转 USB 适配器。 在首次测试或调试测试期间，这种安排尤其实用。

编程设备适配器卡	网络类型	Internet 协议 (IP) 地址	子网掩码
板载适配器卡	连接到工厂 LAN（可能是万维网）	CPU 与编程设备板载适配器卡的网络 ID 必须完全相同。 网络 ID 是 IP 地址的第一部分（前两个八位位组）（例如，211.154.184.16），它决定用户所在的 IP 网络。	CPU 和板载适配器卡的子网掩码必须完全相同。 子网掩码的值通常为 **255.255.255.0**；然而由于您的计算机处于工厂 LAN 中，子网掩码可能有不同的值（例如，**255.255.254.0**）以设置唯一的子网。 子网掩码通过数学 AND 运算同设备 IP 地址组合来确定 IP 子网的边界。
以太网转 USB 适配器卡	连接到独立网络	CPU 与编程设备以太网转 USB 适配器卡的网络 ID 必须完全相同。 网络 ID 是 IP 地址的第一部分（前两个八位位组）（例如，211.154.184.16），它决定用户所在的 IP 网络。	CPU 与编程设备以太网转 USB 适配器卡的子网掩码必须完全相同。 子网掩码的值通常为 **255.255.255.0**。子网掩码通过与设备 IP 地址进行数学 AND 运算来确定 IP 子网的边界。

使用桌面上的“网上邻居”(My Network Places) 分配或检查编程设备的 IP 地址

用户可使用以下菜单选项来分配或检查编程设备的 IP 地址：

- （右键单击）“网上邻居”(My Network Places)
- “属性”(Properties)
- （右键单击）“本地连接”(Local Area Connection)
- “属性”(Properties)

在“本地连接属性”(Local Area Connection Properties) 对话框中，在“此连接使用下列项目:”(This connection uses the following items:) 区域向下滚动到“Internet 协议 (TCP/IP)”(Internet Protocol (TCP/IP))。单击“Internet 协议 (TCP/IP)”(Internet Protocol (TCP/IP))，然后单击“属性”(Properties) 按钮。 选择“自动获得 IP 地址 (DHCP)”(Obtain an IP address automatically (DHCP)) 或“使用下面的 IP 地址”(Use the following IP address)（可输入静态 IP 地址）。

说明

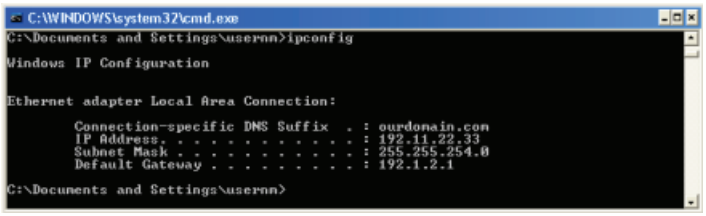
动态主机配置协议 (DHCP, Dynamic Host Configuration Protocol) 通过 DHCP 服务器在编程设备上电时自动为其分配 IP 地址。

使用“ipconfig”和“ipconfig /all”命令检查编程设备的 IP 地址

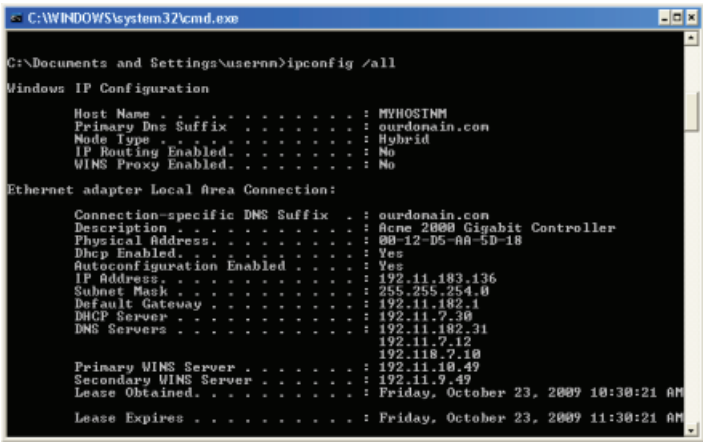
还可以使用以下菜单选项检查编程设备和 IP 路由器（网关）的 IP 地址（如果适用）：

- “开始”(Start) 按钮（在桌面上）
- “运行”(RUN)

在“运行”(Run) 对话框的“打开”(Open) 区域中输入“cmd”，然后单击“确定”(OK) 按钮。 在显示的“C:\WINDOWS\system32\cmd.exe”对话框中，输入命令“ipconfig”。 下面显示了一个结果实例：



使用“ipconfig /all”命令可显示更多信息。 在此可找到编程设备的适配器卡类型和以太网 (MAC) 地址：



为 CPU 分配 IP 地址

可用以下两种方法之一为 CPU 分配 IP 地址：

- 在线分配 IP 地址
- 在项目中组态 IP 地址

7.1.3.2 在线分配 IP 地址

可以在线为网络设备分配 IP 地址。 这在进行初始设备配置时尤其有用。

请按照以下步骤在线分配 IP 地址：

1. 在“项目树”(Project tree) 中，使用以下菜单选项检查是否还没有给 CPU 分配任何 IP 地址：
   - “在线访问”(Online access)
   - <设备所在网络的适配器卡>
   - “更新可访问的设备”(Update accessible devices)

2. 在“项目树”(Project tree) 中，选择以下菜单项：

- “在线访问”(Online access)
- <设备所在网络的适配器卡>
- “更新可访问的设备”(Update accessible devices)
- <设备地址>
- “在线和诊断”(Online & diagnostics)

3. 在“在线和诊断”(Online & diagnostics) 对话框中，选择以下菜单项：

- “功能”(Functions)
- “分配 IP 地址”(Assign IP address)

4. 在“IP 地址”(IP address) 域中，输入新的 IP 地址。

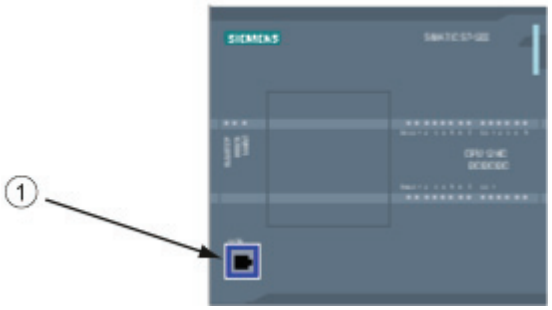
5. 在“项目树”(Project tree) 中，使用以下菜单选项检查新的 IP 地址是否已分配给了 CPU：

- “在线访问”(Online access)
- <设备所在网络的适配器>
- “更新可访问的设备”(Update accessible devices)

7.1.3.3 在项目中组态 IP 地址

组态 PROFINET 接口

使用 CPU 配置机架 (页 245)之后，可组态 PROFINET 接口的参数。 为此，单击 CPU 上的绿色 PROFINET 框以选择 PROFINET 端口。 巡视窗口中的“属性”(Properties) 选项卡会显示 PROFINET 端口。



① PROFINET 端口

组态 IP 地址

以太网 (MAC) 地址： 在 PROFINET 网络中，制造商会为每个设备都分配一个“介质访问控制”地址（MAC 地址）以进行标识。 MAC 地址由六组数字组成，每组两个十六进制数，这些数字用连字符 (-) 或冒号 (:) 分隔并按传输顺序排列（例如 01-23-45-67-89-AB 或 01:23:45:67:89:AB）。

IP 地址： 每个设备也都必须具有一个 Internet 协议 (IP) 地址。 该地址使设备可以在更加复杂的路由网络中传送数据。

每个 IP 地址分为四段，每段占 8 位，并以点分十进制格式表示（例如，211.154.184.16）。 IP 地址的第一部分用于表示网络 ID（您正位于什么网络中？），地址的第二部分表示主机 ID（对于网络中的每个设备都是唯一的）。 IP 地址 192.168.x.y 是一个标准名称，视为未在 Internet 上路由的专用网的一部分。

子网掩码： 子网是已连接的网络设备的逻辑分组。 在局域网 (LAN, Local Area Network) 中，子网中的节点往往彼此之间的物理位置相对接近。 掩码（称为子网掩码或网络掩码）定义 IP 子网的边界。

子网掩码 255.255.255.0 通常适用于小型本地网络。 这就意味着此网络中的所有 IP 地址的前 3 个八位位组应该是相同的，该网络中的各个设备由最后一个八位位组（8 位域）来标识。 举例来说，在小型本地网络中，为设备分配子网掩码 255.255.255.0 和 IP 地址 192.168.2.0 到 192.168.2.255。

不同子网间的唯一连接通过路由器实现。 如果使用子网，则必须部署 IP 路由器。

IP 路由器： 路由器是 LAN 之间的链接。 通过使用路由器，LAN 中的计算机可向其它任何网络发送消息，这些网络可能还隐含着其它 LAN。 如果数据的目的地不在 LAN 内，路由器会将数据转发给可将数据传送到其目的地的另一个网络或网络组。

路由器依靠 IP 地址来传送和接收数据包。

IP 地址属性： 在“属性”(Properties) 窗口中，选择“以太网地址”(Ethernet address) 组态条目。 TIA 门户将显示以太网地址组态对话框，该对话框可将软件项目与接收该项目的 CPU 的 IP 地址相关联。

说明

CPU 不具有预组态的 IP 地址。 必须手动为 CPU 分配 IP 地址。 如果 CPU 连接到网络上的路由器，则也必须输入路由器的 IP 地址。 下载项目时会组态所有 IP 地址。

更多相关信息，请参见“为编程设备和网络设备分配 IP 地址”。

下表定义了 IP 地址的参数：

参数		说明
子网	连接到设备的子网的名称。 单击“添加新子网”(Add new subnet) 按钮以创建新的子网。 默认设置为“未连接”(Not connected)。 有两种连接类型可用： • 默认情况下“未连接”(Not connected) 提供本地连接。 • 网络具有两个或多个设备时，需要子网。	
IP 协议	IP 地址	为 CPU 分配的 IP 地址
	子网掩码	分配的子网掩码
	使用 IP 路由器	单击该复选框以指示 IP 路由器的使用
	路由器地址	为路由器分配的 IP 地址（如果适用）

7.1.4 测试 PROFINET 网络

在完成组态后，下载项目到 CPU 中。 下载项目时会组态所有 IP 地址。

在线为设备分配 IP 地址

S7-1200 CPU 不具有预组态的 IP 地址。 必须手动为 CPU 分配 IP 地址。

要在线给设备分配 IP 地址，请参考“在线分配 IP 地址”以了解此过程的逐步操作信息。

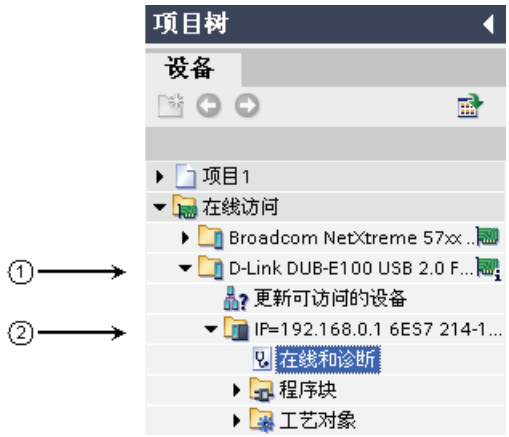
要在项目中分配 IP 地址，必须在设备配置中组态 IP 地址，保存配置并将其下载到 PLC。 更多相关信息，请参见“为项目组态 IP 地址”。

说明

如果已在线分配 IP 地址，则可采用在线或离线硬件配置方法更改在线分配的 IP 地址。

如果已在离线硬件配置期间分配了 IP 地址，则只能采用离线硬件配置方法更改项目中分配的 IP 地址。

请使用“在线访问”(Online access) 显示所连接的 CPU 的 IP 地址，如下所示。



① 该编程设备上两个以太网网络中的第二个网络

② 该以太网网络中唯一的 S7-1200 CPU 的 IP 地址

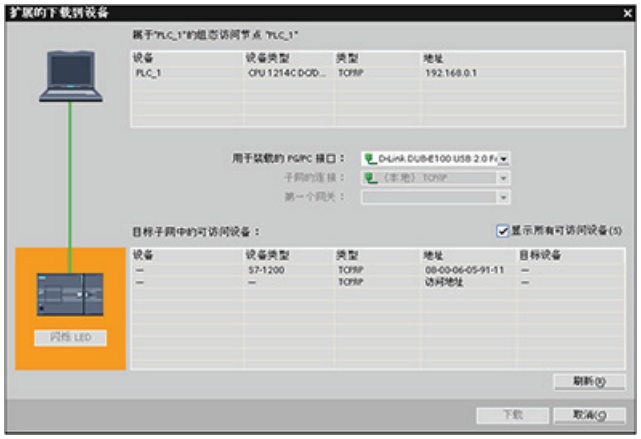
说明

编程设备的所有组态网络都将显示。 必须选择正确的网络才能显示所需的 S7-1200 CPU 的 IP 地址。

使用“扩展的下载到设备”(Extended download to device) 对话框测试所连接的网络设备

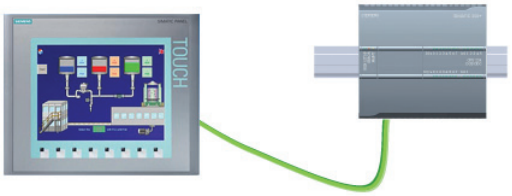
S7-1200 CPU“下载到设备”(Download to device) 功能及其“扩展的下载到设备”(Extended download to device) 对话框可以显示所有可访问的网络设备，以及是否为所有设备都分

配了唯一的 IP 地址。 要显示全部可访问和可用的设备以及为其分配的 MAC 和 IP 地址，请选中“显示所有可访问设备”(Show all accessible devices) 复选框。



如果所需网络设备不在此列表中，则说明由于某种原因而中断了与该设备的通信。 必须检查设备和网络是否有硬件和/或组态错误。

7.2 HMI 到 PLC 通信



CPU 支持通过 PROFINET 端口与 HMI 通信。 设置 CPU 和 HMI 之间的通信时必须考虑以下要求：

组态/设置：

- 必须组态 CPU 的 PROFINET 端口与 HMI 连接。
- 必须已设置和组态 HMI。
- HMI 组态信息是 CPU 项目的一部分，可以在项目内部进行组态和下载。
- 一对一通信不需要以太网交换机；网络中有两个以上的设备时需要以太网交换机。

说明

安装在机架上的 Siemens CSM1277 4 端口以太网交换机可用于连接 CPU 和 HMI 设备。 CPU 上的 PROFINET 端口不包含以太网交换设备。

支持的功能：

- HMI 可以对 CPU 读/写数据。
- 可基于从 CPU 重新获取的信息触发消息。
- 系统诊断

说明

WinCC Basic 和 STEP 7 Basic 是 TIA 门户的组件。 有关组态 HMI 的更多信息，请参见 WinCC Basic。

组态 HMI 与 CPU 之间的通信时所需的步骤

步骤	任务
1	建立硬件通信连接 通过 PROFINET 接口建立 HMI 和 CPU 之间的物理连接。由于 CPU 内置了自动跨接功能，所以对该接口既可以使用标准以太网电缆，又可以使用跨接以太网电缆。 连接一个 HMI 和一个 CPU 不需要以太网交换机。 更多相关信息，请参见“与编程设备通信：建立硬件通信连接 (页 245)”。
2	配置设备 更多相关信息，请参见“与编程设备通信：组态设备 (页 245)”。
3	组态 HMI 与 CPU 之间的逻辑网络连接 更多相关信息，请参见“HMI 与 PLC 通信：组态 HMI 和 CPU 之间的逻辑网络连接 (页 257)”。
4	在项目中组态 IP 地址 使用相同的组态过程；但必须为 HMI 和 CPU 组态 IP 地址。 更多相关信息，请参见“与编程设备通信：在项目中组态 IP 地址 (页 251)”。
5	测试 PROFINET 网络 必须为每个 CPU 都下载相应的组态。 更多相关信息，请参见“与编程设备通信：测试 PROFINET 网络 (页 253)”。

7.2.1 组态 HMI 与 CPU 之间的逻辑网络连接

使用 CPU 配置机架后，您即准备好组态网络连接。

在“设备和网络”(Devices and Networks) 门户中，使用“网络视图”(Network view) 创建项目中各设备之间的网络连接。 要创建以太网连接，请选择 CPU 上的绿色（以太网）框。拖出一条线连接到 HMI 设备上的以太网框。 释放鼠标按钮，即可创建以太网连接。

操作	结果
选择“网络视图”(Network view) 以显示要连接的设备。	
选择一个设备上的端口，然后将连接拖到第二个设备上的端口上。	
释放鼠标按钮以创建网络连接。	

7.3 PLC 到 PLC 通信

通过使用 TSEND_C 和 TRCV_C 指令，一个 CPU 可与网络中的另一个 CPU 进行通信。

设置两个 CPU 之间的通信时必须考虑以下事宜：

- 组态/设置： 需要进行硬件配置。
- 支持的功能： 向对等 CPU 读/写数据
- 一对一通信不需要以太网交换机；网络中有两个以上的设备时需要以太网交换机。

组态两个 CPU 之间的通信时所需的步骤

步骤	任务
1	建立硬件通信连接 通过 PROFINET 接口建立两个 CPU 之间的物理连接。由于 CPU 内置了自动跨接功能，所以对该接口既可以使用标准以太网电缆，又可以使用跨接以太网电缆。 连接两个 CPU 时不需要以太网交换机。 更多相关信息，请参见“与编程设备通信：建立硬件通信连接”。
2	配置设备 必须组态两个项目，其中每个项目有一个 CPU。 更多相关信息，请参见“与编程设备通信：组态设备”。
3	组态两个 CPU 之间的逻辑网络连接 更多相关信息，请参见“组态两个 CPU 之间的通信：组态两个 CPU 之间的逻辑网络连接 (页 259)”。
4	在项目中组态 IP 地址 使用相同的组态过程；但必须为两个 CPU（例如，PLC_1 和 PLC_2）组态 IP 地址。 更多相关信息，请参见“与编程设备通信：在项目中组态 IP 地址”。
5	组态传送（发送）和接收参数 必须在两个 CPU 中均组态 TSEND_C 和 TRCV_C 指令，才能实现两个 CPU 之间的通信。 更多相关信息，请参见“组态两个 CPU 之间的通信：组态传送（发送）和接收参数 (页 259)”。
6	测试 PROFINET 网络 必须为每个 CPU 都下载相应的组态。 更多相关信息，请参见“组态编程设备与 CPU 之间的通信：测试 PROFINET 网络”。

7.3.1 组态两个 CPU 之间的逻辑网络连接

使用 CPU 配置机架后，您即准备好组态网络连接。

在“设备和网络”(Devices and Networks) 门户中，使用“网络视图”(Network view) 创建项目中各设备之间的网络连接。 要创建 PROFINET 连接，请选择第一个 PLC 上的绿色 (PROFINET) 框。 拖出一条线连接到第二个 PLC 上的 PROFINET 框。 释放鼠标按钮，即可创建 PROFINET 连接。

操作	结果
选择“网络视图”(Network view) 以显示要连接的设备。	
选择一个设备上的端口，然后将连接拖到第二个设备上的端口上。	
释放鼠标按钮以创建网络连接。	

7.3.2 组态传送（发送）和接收参数

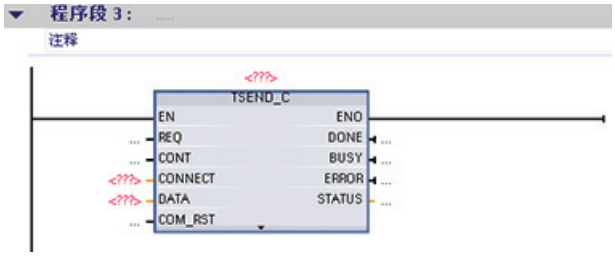
传输块 (T-block) 通信用于建立两个 CPU 之间的连接。 在 CPU 可进行 PROFINET 通信前，必须组态传送（或发送）消息和接收消息的参数。 这些参数决定了在向目标设备传送消息或从目标设备接收消息时的通信工作方式。

7.3.2.1 组态 TSEND_C 指令传送（发送）参数

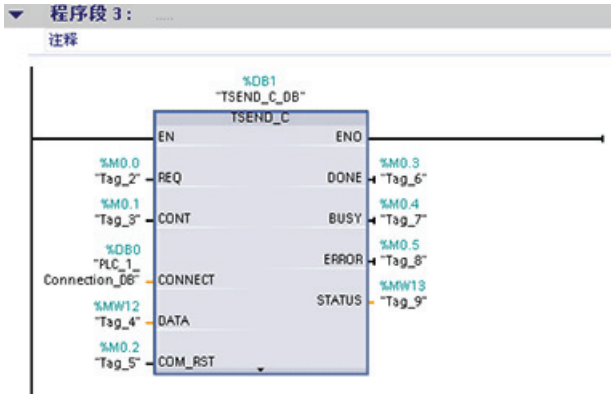
TSEND_C 指令

TSEND_C 指令 (页 175)可创建与伙伴站的通信连接。通过该指令可设置和建立连接，并会在通过指令断开连接前一直自动监视该连接。 TSEND_C 指令兼具 TCON、TDISCON 和 TSEND 指令的功能。

通过 STEP 7 Basic 中的设备配置，可以组态 TSEND_C 指令传送数据的方式。 首先，从“通信”(Communications) 文件夹的“扩展指令”(Extended Instructions) 中将该指令插入程序中。 该指令将与“调用选项”(Call options) 对话框一起显示，在该对话框中可以分配用于存储 TSEND_C 指令参数的 DB。



可以为输入和输出分配变量存储位置，如下图所示。



组态常规参数

在 TSEND_C 指令的“属性”(Properties) 组态对话框中指定通信参数。 只要选中了 TSEND_C 指令的任何一部分，此对话框就会出现在页面底部附近。

组态连接参数

每个 CPU 都集成了一个支持标准 PROFINET 通信的 PROFINET 端口。在以下两种连接类型中描述了支持的以太网协议：

协议	协议名称	用途
RFC 1006	ISO on TCP	消息的分割和重组
TCP	传输控制协议	帧传输

ISO on TCP (RFC 1006)

ISO on TCP 是一种能够将 ISO 应用移植到 TCP/IP 网络的机制。该协议有以下特点：

- 它是与硬件关系紧密的高效通信协议
- 它适合用于中等大小或较大的数据量（最多 8192 字节）
- 与 TCP 相比，它的消息提供了数据结束标识符并且它是面向消息的。
- 具有路由功能；可用于 WAN
- 可用于实现动态数据长度。
- 由于使用 SEND/RECEIVE 编程接口的缘故，需要对数据管理进行编程。

通过传输服务访问点 (TSAP, Transport Service Access Point)，TCP 协议允许有多个连接访问单个 IP 地址（最多 64K 个连接）。借助 RFC 1006，TSAP 可唯一标识连接到同一个 IP 地址的这些通信端点连接。

在“连接参数”(Connection Parameters) 对话框的“地址详细信息”(Address Details) 部分，定义要使用的 TSAP。在“本地 TSAP”(Local TSAP) 域中输入 CPU 中连接的 TSAP。在“伙伴 TSAP”(Partner TSAP) 域下输入为伙伴 CPU 中的连接分配的 TSAP。

参数	定义
常规	
端点：伙伴	分配给伙伴（接收）CPU 的名称
接口	分配给接口的名称
子网	分配给子网的名称
地址	分配的 IP 地址
连接类型	以太网协议的类型
连接 ID	ID 号
连接数据	本地和伙伴 CPU 的数据存储位置
主动建立连接	选择本地或伙伴 CPU 作为主动连接方的单选按钮
地址详细信息	
TSAP[1] (ASCII)	ASCII 格式的本地和伙伴 CPU TSAP
TSAP ID	十六进制格式的本地和伙伴 CPU TSAP

1 组态与 S7-1200 CPU 的 ISO-on-TCP 连接时，请在被动通信伙伴的 TSAP 扩展中仅使用 ASCII 字符。

传输控制协议 (TCP)

TCP 是由 RFC 793 描述的一种标准协议：传输控制协议。TCP 的主要用途是在过程对之间提供可靠、安全的连接服务。该协议有以下特点：

- 由于它与硬件紧密相关，因此它是一种高效的通信协议
- 它适合用于中等大小或较大的数据量（最多 8192 字节）
- 它为应用带来了更多的便利，特别是：
  - 错误恢复
  - 流控制
  - 可靠性
- 它是一种面向连接的协议
- 它可以非常灵活地用于只支持 TCP 的第三方系统
- 有路由功能
- 只能应用静态数据长度。
- 消息会被确认。
- 使用端口号对应用程序寻址。
- 大多数用户应用协议（例如 TELNET 和 FTP）都使用 TCP。
- 由于使用 SEND/RECEIVE 编程接口的缘故，需要对数据管理进行编程。

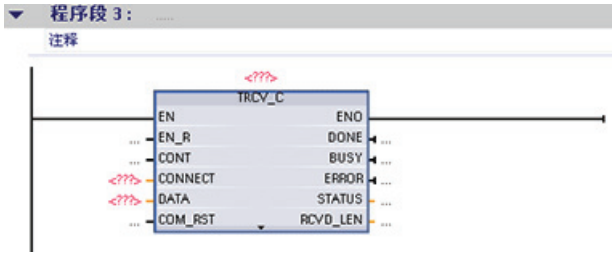
参数	定义
常规	
端点： 伙伴	分配给伙伴（接收）CPU 的名称
接口	分配给接口的名称
子网	分配给子网的名称
地址	分配的 IP 地址
连接类型	以太网协议的类型
连接 ID	ID 号
连接数据	本地和伙伴 CPU 的数据存储位置
主动建立连接	选择本地或伙伴 CPU 作为主动连接方的单选按钮
地址详细信息	
端口（十进制）	十进制格式的伙伴 CPU 端口

7.3.2.2 组态 TRCV_C 指令接收参数

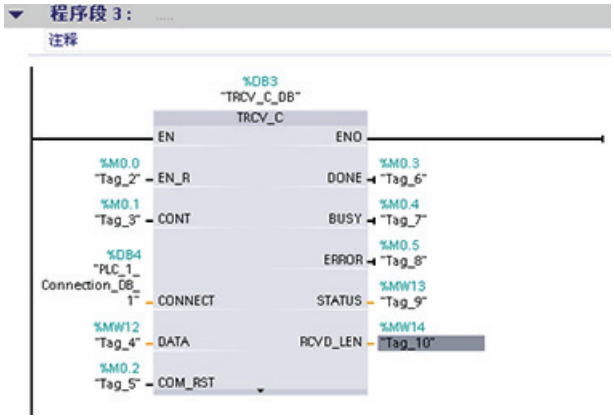
TRCV_C 指令

TRCV_C 指令 (页 175)可创建与伙伴站的通信连接。通过该指令可设置和建立连接，并会在通过指令断开连接前一直自动监视该连接。 TRCV_C 指令兼具 TCON、TDISCON 和 TRCV 指令的功能。

通过 STEP 7 Basic 中的 CPU 组态，可以组态 TRCV_C 指令接收数据的方式。 首先，从“通信”(Communications) 文件夹的“扩展指令”(Extended Instructions) 中将该指令插入程序中。 该指令将与“调用选项”(Call options) 对话框一起显示，在该对话框中可以分配用于存储 TRCV_C 指令参数的 DB。



可以为输入和输出分配变量存储位置，如下图所示。



组态常规参数

在 TRCV_C 指令的“属性”(Properties) 组态对话框中指定通信参数。 只要选中了 TRCV_C 指令的任何一部分，此对话框就会出现在页面底部附近。

组态连接参数

每个 CPU 都集成了一个支持标准 PROFINET 通信的 PROFINET 端口。在以下两种连接类型中描述了支持的以太网协议：

协议	协议名称	用途
RFC 1006	ISO on TCP	消息的分割和重组
TCP	传输控制协议	帧传输

ISO on TCP (RFC 1006)

ISO on TCP 是一种能够将 ISO 应用移植到 TCP/IP 网络的机制。该协议有以下特点：

- 它是与硬件关系紧密的高效通信协议
- 它适合用于中等大小或较大的数据量（最多 8192 字节）
- 与 TCP 相比，它的消息提供了数据结束标识符并且它是面向消息的。
- 具有路由功能；可用于 WAN
- 可用于实现动态数据长度。
- 由于使用 SEND/RECEIVE 编程接口的缘故，需要对数据管理进行编程。

通过传输服务访问点 (TSAP, Transport Service Access Point)，TCP 协议允许有多个连接访问单个 IP 地址（最多 64K 个连接）。借助 RFC 1006，TSAP 可唯一标识连接到同一个 IP 地址的这些通信端点连接。

在“连接参数”(Connection Parameters) 对话框的“地址详细信息”(Address Details) 部分，定义要使用的 TSAP。在“本地 TSAP”(Local TSAP) 域中输入 CPU 中连接的 TSAP。在“伙伴 TSAP”(Partner TSAP) 域下输入为伙伴 CPU 中的连接分配的 TSAP。

参数	定义
常规	
端点：伙伴	分配给伙伴（接收）CPU 的名称
接口	分配给接口的名称
子网	分配给子网的名称
地址	分配的 IP 地址
连接类型	以太网协议的类型
连接 ID	ID 号
连接数据	本地和伙伴 CPU 的数据存储位置

参数	定义
主动建立连接	选择本地或伙伴 CPU 作为主动连接方的单选按钮
地址详细信息	
TSAP[1] (ASCII)	ASCII 格式的本地和伙伴 CPU TSAP
TSAP ID	十六进制格式的本地和伙伴 CPU TSAP

1 组态与 S7-1200 CPU 的 ISO-on-TCP 连接时，请在被动通信伙伴的 TSAP 扩展中仅使用 ASCII 字符。

传输控制协议 (TCP)

TCP 是由 RFC 793 描述的一种标准协议：传输控制协议。TCP 的主要用途是在过程对之间提供可靠、安全的连接服务。该协议有以下特点：

- 由于它与硬件紧密相关，因此它是一种高效的通信协议
- 它适合用于中等大小或较大的数据量（最多 8192 字节）
- 它为应用带来了更多的便利，特别是：
  - 错误恢复
  - 流控制
  - 可靠性
- 它是一种面向连接的协议
- 它可以非常灵活地用于只支持 TCP 的第三方系统
- 有路由功能
- 只能应用静态数据长度。
- 消息会被确认。
- 使用端口号对应用程序寻址。
- 大多数用户应用协议（例如 TELNET 和 FTP）都使用 TCP。
- 由于使用 SEND/RECEIVE 编程接口的缘故，需要对数据管理进行编程。

参数	定义
常规	
端点： 伙伴	分配给伙伴（接收）CPU 的名称
接口	分配给接口的名称
子网	分配给子网的名称
地址	分配的 IP 地址
连接类型	以太网协议的类型
连接 ID	ID 号
连接数据	本地和伙伴 CPU 的数据存储位置
主动建立连接	选择本地或伙伴 CPU 作为主动连接方的单选按钮
地址详细信息	
端口（十进制）	十进制格式的本地 CPU 端口

7.4 引用信息

7.4.1 查找 CPU 上的以太网 (MAC) 地址

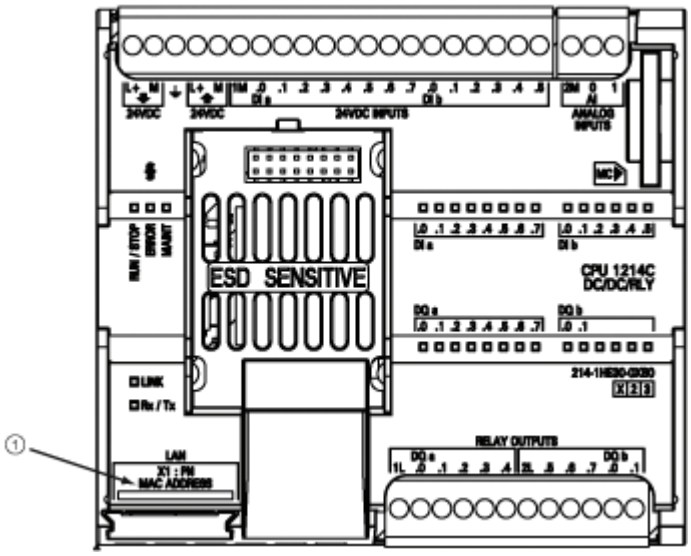
在 PROFINET 网络中，“介质访问控制”地址（MAC 地址）是指制造商为了标识适配器卡而分配的标识符。 MAC 地址通常用制造商的注册标识号进行编码。

外观良好、按标准 (IEEE 802.3) 格式印制的 MAC 地址由六组数字组成，每组两个十六进制数，这些数字组用连字符 (-) 或冒号 (:) 分隔并按传输顺序排列（例如 01-23-45-67-89-ab 或 01:23:45:67:89:ab）。

说明

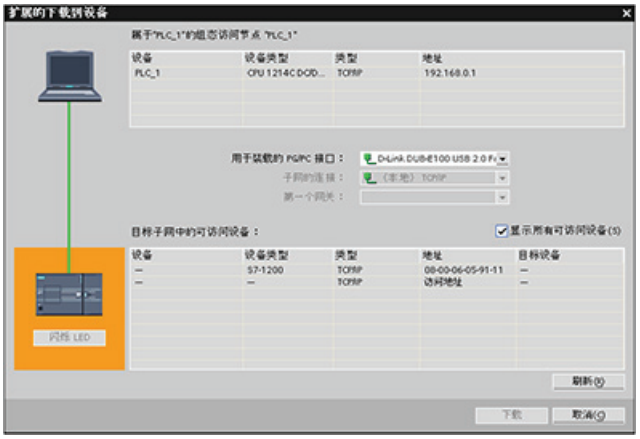
每个 CPU 在出厂时都已装载了一个永久、唯一的 MAC 地址。 您无法更改 CPU 的 MAC 地址。

MAC 地址印在 CPU 正面左下角位置。 请注意，必须提起下面的 TB 门才能看到 MAC 地址信息。



① MAC 地址

最初，CPU 没有 IP 地址，只有工厂安装的 MAC 地址。 PROFINET 通信要求为所有设备都分配唯一的 IP 地址。



可以使用 CPU“下载到设备”(Download to device) 功能及其“扩展的下载到设备”(Extended download to device) 对话框，显示所有可访问的网络设备以确保已经为所有设备分配了唯一的 IP 地址。此对话框可显示所有可访问和可用的设备以及所分配的 MAC 和 IP 地址。 在识别缺少所需唯一 IP 地址的设备时，MAC 地址就十分重要。

7.4.2 组态网络时间协议同步

网络时间协议 (NTP, Network Time Protocol) 被广泛用于使计算机系统的时钟与 Internet 时间服务器同步。 它在 LAN 上可实现的时间精度通常小于 1 毫秒，而在 WAN 上通常可达几毫秒。 典型的 NTP 组态采用多个冗余服务器和多种不同的网络路径，以获得高精度和可靠性。

NTP 子网按层级方式构成，其中每一级都分配有一个称为层的编号。 最底一级的层 1（主）服务器直接与国家时间服务同步。 下一个较高级的层 2（辅）服务器与层 1 服务器同步，依此类推。

时间同步参数

在“属性”(Properties) 窗口中，选择“时间同步”(Time synchronization) 组态条目。 TIA 门户将显示“时间同步”(Time synchronization) 组态对话框：

说明

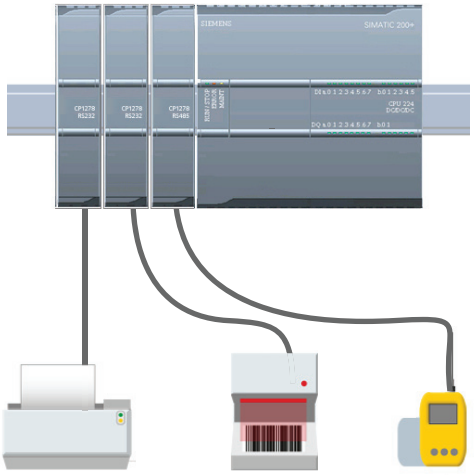
下载项目时会组态所有 IP 地址。

下表定义了时间同步的参数：

参数	定义
启用使用网络时间协议 (NTP) 服务器的日时钟同步 (Enable time-of-day synchronization using Network Time Protocol (NTP) servers)	单击该复选框可启用使用 NTP 服务器的日时钟同步。
服务器 1 (Server 1)	为网络时间服务器 1 分配的 IP 地址
服务器 2 (Server 2)	为网络时间服务器 2 分配的 IP 地址
服务器 3 (Server 3)	为网络时间服务器 3 分配的 IP 地址
服务器 4 (Server 4)	为网络时间服务器 4 分配的 IP 地址
时间同步间隔 (Time synchronization interval)	时间间隔值（秒）

8 点对点 (PtP) 通信

CPU 支持使用点对点协议 (PtP) 进行基于字符的串行通信，在该通信中，通过用户应用程序完全定义和实施所选的协议。 PtP 可提供最大的自由度和灵活性，但需要在用户程序中包含大量的实现。



PtP 可用于实现多种可能性：

- 能够将信息直接发送到外部设备，例如，打印机
- 能够从其它设备（例如，条码阅读器、RFID 阅读器、第三方照相机或视觉系统以及许多其它类型的设备）接收信息
- 能够与其它设备（例如，GPS 设备、第三方照相机或视觉系统、无线调制解调器以及更多其它设备）交换信息（发送和接收数据）

PtP 通信属于串行通信，它使用标准 UART 来支持多种波特率和奇偶校验选项。 RS232 或 RS485 通信模块 (CM) 提供了用于执行 PtP 通信的电气接口。

STEP 7 Basic 提供了指令库，可用来针对您的应用进行编程。 这些库可为以下协议提供 PtP 通信功能：

- USS 驱动协议
- Modbus RTU 主站协议
- Modbus RTU 从站协议

8.1 使用 RS232 和 RS485 通信模块

以下两种通信模块 (CM) 提供有 PtP 通信接口： CM 1241 RS485 (页 369) 和 CM 1241 RS232 (页 371)。 最多可以连接 3 个 CM（类型不限）。 请将 CM 安装在 CPU 或另一个 CM 的左侧。有关模块安装和拆卸的详细说明，请参考“安装”一章 (页 31)。

RS232 和 RS485 通信模块有以下特征：

- 端口经过隔离
- 支持点对点协议
- 通过扩展指令和库功能进行组态和编程
- 通过 LED 显示传送和接收活动
- 显示诊断 LED
- 由 CPU 供电。 不必连接外部电源。

请参考通信模块的技术规范 (页 369)。

8.2 组态通信端口

可以通过以下两种方式组态通信模块：

- 使用 STEP 7 Basic 中的设备配置组态端口参数（波特率和奇偶校验）、发送参数和接收参数。 设备配置设置永久存储在 CPU 中。 在循环上电和从 RUN 模式切换到 STOP 模式后会应用这些设置。
- 使用 PORT_CFG、SEND_CFG 和 RCV_CFG 指令设置参数。 这些指令设置的端口设置在 CPU 处于 RUN 模式期间有效。 在切换到 STOP 模式或循环上电后，这些端口设置会恢复为设备配置设置。

配置硬件设备 (页 75)之后，通过选择机架上的其中一个 CM 来组态通信接口的参数。

巡视窗口的“属性”(Properties) 选项卡将显示所选 CM 的参数。请选择“端口组态”(Port configuration) 编辑以下参数：

- 波特率
- 奇偶校验
- 停止位的数目
- 流控制（仅限 RS232）
- 等待时间

无论是组态 RS232 还是 RS485 通信模块，除流控制外，其它端口组态参数都是相同的。 但是，参数值可能不同。

还可以在用户程序中使用 PORT_CFG (页 288) 指令对端口进行组态（或更改现有组态）。

说明

在用户程序中通过 PORT_CFG 指令设置的参数值会覆盖通过 STEP 7 Basic 设置的端口组态设置。 请注意，发生掉电时，S7-1200 不会保留通过 PORT_CFG 指令设置的参数。

波特率 (Baud rate)： 波特率的默认值为 9.6 Kbps。有效选项有：

300 波特	2.4 kb	19.2 kb	76.8 kb
600 波特	4.8 kb	28.4 kb	115.2 kb
1.2 kb	9.6 kb	57.6 kb	

奇偶校验 (Parity)： 奇偶校验的默认值是无奇偶校验。有效选项有：

- 无奇偶校验
- 偶校验
- 奇校验
- 传号校验（奇偶校验位始终设置为 1）
- 空号校验（奇偶校验位始终设置为 0）

停止位的数目 (Number of stop bits)： 停止位的数目可以是 1 或 2。 默认值是 1。

流控制 (Flow control)： 对于 RS232 通信模块，可以选择硬件或软件流控制，如“管理流控制 (页 275)”部分所述。 如果选择硬件流控制，则可以选择是 RTS 信号始终激活还是切换 RTS。 如果选择软件流控制，则可以为 XON 和 XOFF 字符定义 ASCII 字符。

RS485 通信模块不支持流控制。

等待时间 (Wait time)： 等待时间是指通信模块在声明 RTS 后等待接收 CTS 的时间或者在接收 XOFF 后等待接收 XON 的时间，具体取决于流控制类型。 如果在通信模块接收到预期的 CTS 或 XON 之前超过了等待时间，通信模块将中止传送操作并向用户程序返回错误。 指定等待时间，以毫秒表示。 范围是 0 到 65535 毫秒。

8.3 管理流控制

流控制是指为了不丢失数据而用来平衡数据发送和接收的一种机制。 流控制可确保传送设备发送的信息量不会超出接收设备所能处理的信息量。 流控制可以通过硬件或软件来

实现。 RS232 CM 支持硬件及软件流控制。 RS485 CM 不支持流控制。 在组态端口 (页 274)时或使用 PORT_CFG 指令指定流控制类型。

硬件流控制通过请求发送 (RTS, Request To Send) 和允许发送 (CTS, Clear To Send) 通信信号来实现。 对于 RS232 CM，RTS 信号从引脚 7 输出，而 CTS 信号通过引脚 8 接收。CM 1241 是 DTE（Data Terminal Equipment，数据终端设备）设备，其将 RTS 声明为输出并将 CTS 作为输入来监视。

硬件流控制：RTS 切换

如果为 RS232 CM 启用 RTS 切换的硬件流控制，则模块会将 RTS 信号设置为激活状态以发送数据。 它还会监视 CTS 信号以确定接收设备是否能接收数据。 CTS 信号激活后，只要 CTS 信号保持激活状态，模块便可发送数据。 如果 CTS 信号变为非激活状态，则传送必须停止。

CTS 信号变为激活状态时，传送会继续执行。 如果 CTS 信号在组态的等待时间内未激活，则模块会中止传送并向用户程序返回错误。 在端口组态 (页 274)中指定等待时间。

对于需要“传送已激活”信号的设备，适合使用 RTS 切换流控制。 例如，无线调制解调器使用 RTS 作为“键”信号来激励无线发送器。 RTS 切换流控制对于标准电话调制解调器不起作用。 对电话调制解调器使用“RTS 始终激活”选项。

硬件流控制： RTS 始终激活

在“RTS 始终激活”节点中，CM 1241 默认情况下将 RTS 设置为激活状态。 设备（如电话调制解调器等）监视来自 CM 的 RTS 信号，并将该信号用作允许发送信号。 调制解调器仅在 RTS 处于激活状态时才向 CM 传送数据，即，电话调制解调器在见到激活的 CTS 信号后发送数据。如果 RTS 处于非激活状态，电话调制解调器不向 CM 传送数据。

要使调制解调器随时都能向 CM 发送数据，请组态“RTS 始终激活”硬件流控制。 CM 因此会将 RTS 信号设置为始终激活。 即使模块无法接受字符，CM 也不会将 RTS 设置为非激活状态。 传送设备必须确保不会使 CM 的接收缓冲区超负荷运行。

利用数据终端就绪 (DTR) 和数据设备就绪 (DSR) 信号

对于这两种硬件流控制类型的任何一种，CM 都会将 DTR 设置为激活状态。 只有当 DSR 信号变为激活状态时，模块才会进行传送。 仅在发送操作开始时评估 DSR 的状态。 如果 DSR 在传送操作开始后变为非激活状态，将不能暂停传送操作。

软件流控制

软件流控制使用消息中的特殊字符来实现流控制。 这些字符是表示 XON 和 XOFF 的 ASCII 字符。

XOFF 指示传送必须停止。 XON 指示传送可以继续。

传送设备从接收设备收到 XOFF 字符时，将停止传送。 传送设备收到 XON 字符时，传送又继续进行。 如果 CM 在通过端口组态 (页 274)指定的等待时间内没有收到 XON 字符，它将中止传送并向用户程序返回错误。

软件流控制需要全双工通信，因为在传送过程中接收伙伴必须能够将 XOFF 发送到传送伙伴。 软件流控制只能用于仅包含 ASCII 字符的消息。 二进制协议无法使用软件流控制。

8.4 组态传送（发送）和接收参数

在 PLC 可进行 PtP 通信前，必须组态传送（或发送）消息和接收消息的参数。 这些参数决定了在向目标设备传送消息或从目标设备接收消息时的通信工作方式。

组态传送（发送）参数

组态 CM 期间，通过为所选 CM 指定“传送消息组态”(Transmit message configuration) 属性，可组态通信接口传送数据的方式。

还可以使用 SEND_CFG (页 290) 指令，通过用户程序动态组态或更改传送消息参数。

说明

在用户程序中通过 SEND_CFG 指令设置的参数值会覆盖该端口组态设置。 请注意，发生掉电时，CPU 不会保留通过 SEND_CFG 指令设置的参数。

参数	定义
RTS 接通延时 (RTS On delay)	指定在 RTS 激活后传送启动前需等待的时间。 范围为 0 到 65535 ms，默认值为 0。仅当端口组态 (页 274)指定的是硬件流控制时，该参数才有效。 在经过 RTS 接通延迟时间后才会评估 CTS。 该参数仅适用于 RS232 模块。
RTS 关断延时 (RTS Off delay)	指定传送完成后 RTS 禁用前需等待的时间。 范围为 0 到 65535 ms，默认值为 0。仅当端口组态 (页 274)指定的是硬件流控制时，该参数才有效。 该参数仅适用于 RS232 模块。
在消息开始时发送中断 (Send break at message start) 中断期间的位时间数 (Number of bit times in a break)	指定在每条消息开始时，在 RTS 接通延时（如果已组态）已到且 CTS 已激活的情况下先发送中断。 用户指定多少个位的时间构成一个中断，线路在中断期间保持空号状态。 默认值为 12，最大值为 65535，即最长 8 秒的限制。
发送中断后线路空闲信号 (Send idle line after a break) 中断后线路空闲 (Idle line after a break)	指定在消息开始的中断后发送线路空闲信号。 “中断后线路空闲”(Idle line after a break) 参数指定多少个位时间构成一次线路空闲，线路在空闲期间保持传号状态。 默认值为 12，最大值为 65535，即最长 8 秒的限制。

组态接收参数

通过设备配置，可以组态通信接口接收数据以及识别消息开始和结束的方式。 在所选 CM 的接收消息组态中指定以上参数。

还可以使用 RCV_CFG (页 292) 指令，通过用户程序动态组态或更改接收消息参数。

说明

在用户程序中通过 RCV_CFG 指令设置的参数值会覆盖该端口组态设置。 请注意，发生掉电时，CPU 不会保留通过 RCV_CFG 指令设置的参数。

更多相关信息，请参阅 RCV_CFG 指令。

消息开始参数

用户可以决定通信模块识别消息开始的方式。 在满足所组态的结束条件之前，开始字符以及组成消息的字符会一直进入接收缓冲区。

可以指定多个开始条件。 只有满足所有开始条件后，才视为消息开始。 例如，如果用户组态了空闲线路时间和特定开始字符，CM 将首先查找要满足的空闲线路时间要求，然后查找指定的开始字符。 如果收到一些其它字符而不是指定的开始字符，CM 将通过再次查找空闲线路时间来重新启动消息开始条件搜索。

检查开始条件的顺序是：

- 线路空闲
- 线路中断
- 字符或字符序列

检查多个开始条件时，如果有一个条件没有满足，则 CM 将从第一个所需的条件开始重新启动检查。

参数	定义
“开始字符”字符	“开始字符”条件指定在成功接收到特定字符时开始消息传输。该字符将是消息中的第一个字符。 在该特定字符前接到的任何字符都将被丢弃。
以任意字符开始	“任意字符”条件指定成功接收的任何字符都将导致消息开始。该字符将是消息中的第一个字符。
线路中断	“线路中断”条件指定应在接收中断字符后开始消息接收操作。
线路空闲	“线路空闲”条件指定在接收线路空闲或空闲了指定位时间后开始消息接收操作。一旦出现该条件，就会导致消息开始。

参数	定义
特殊条件： 通过单个字符识别消息开始	指定通过特殊字符指示消息开始。 默认值是 STX。
特殊条件： 通过字符序列识别消息开始 (Recognize message start with a character sequence)	指定通过特殊字符序列指示消息开始。 可以为每个序列最多指定 5 个字符。 对于每个字符位置，可以指定一个具体的十六进制字符，或者指定在序列匹配时忽略该字符。 程序将根据组态的开始条件对进入序列进行评估，直到满足开始条件为止。 只要满足了开始序列，就会开始评估结束条件。 最多可以组态 5 个特定的字符序列，用户可以根据需要启用或禁用这些字符序列。 只要有一个组态的字符序列出现，就表示满足该开始条件。
配置示例	 对于该组态，只要出现其中一个序列，即会满足开始条件： • 接到一个由五个字符构成的序列，且其第一个字符是 0x6A 而第五个字符是 0x1C 时。 对于该组态，位置 2、3 和 4 的字符可以是任意字符。 在接到第五个字符后，将开始评估结束条件。 • 接到两个连续的 0x6A 字符（前面为任意字符）时。 在这种情况下，会在接到第二个 0x6A 后开始评估结束条件（3 个字符）。 第一个 0x6A 前面的字符包含在开始条件中。 满足该开始条件的实例序列有： • <任意字符> 6A 6A • 6A 12 14 18 1C • 6A 44 A5 D2 1C

消息结束参数

用户还可以组态通信接口识别消息结束的方式。 可以组态多个消息结束条件。 如果出现组态条件中的任何一个，消息就会结束。

可同时指定多个结束条件。 只要满足其中一个结束条件，消息就会结束。 例如，可以采用消息超时 300 ms、字符间超时 40 个位的时间以及最大长度 50 个字节作为消息结束的结束条件。 如果接收消息的时间超过 300 ms、任意两个字符间的间隔超过 40 个位的时间或接收到 50 个字节，消息即会结束。

参数	定义
通过消息超时识别消息结束 (Recognize message end by message timeout)	经过了组态的消息结束等待时间后，视为消息结束。 消息超时时间从接到符合消息开始条件的第一个字符时开始计算。 默认值是 200 ms，有效范围是 0 到 65535 ms。
通过响应超时识别消息结束 (Recognize message end by response timeout)	如果在接收到有效的开始序列之前超过了组态的响应等待时间，视为消息结束。 响应超时时间从传送结束时开始计算。 默认响应超时是 200 ms，有效范围是 0 到 65535 ms。用户必须组态另一个结束条件来指示实际的消息结束。
通过字符间隙识别消息结束 (Recognize message end by inter-character gap)	经过了组态的消息中两个连续字符间的最大超时后，视为消息结束。 字符间隙的默认值是 12 个位的时间，最大值是 65535 个位的时间，即最长 8 秒。
通过最大长度识别消息结束 (Recognize message end by max length)	在接收到组态的最大字符数后，视为消息结束。 默认值是 0 字节，最大值是 1024 字节。
从消息读取消息长度 (Read message length from message)	消息本身指定消息长度。 在接收到指定长度的消息后，视为消息结束。 以下说明了用于指定和解释消息长度的方法。
通过字符识别消息结束 (Recognize message end with a character)	在接收到指定的字符后，视为消息结束。

参数	定义
通过字符序列识别消息结束 (Recognize message end with a character sequence)	在接收到指定的字符序列后，视为消息结束。 可以指定最多由 5 个字符组成的序列。 对于每个字符位置，可以指定一个具体的十六进制字符，或者指定在序列匹配时忽略该字符。 结束条件不包括被忽略的前导字符。 结束条件包括被忽略的尾随字符。
配置示例	 在这种情况下，在接收到两个连续的 0x7A 字符（后跟任意两个字符）时，即满足结束条件。 0x7A 0x7A 序列前面的字符不是结束字符序列的组成部分。 终止结束字符序列时需要跟在 0x7A 0x7A 序列后面的两个字符。 字符位置 4 和 5 的值不相关，但必须接收它们才能满足结束条件。

在消息中指定消息长度

选择在消息中包括消息长度这一特殊条件时，必须提供三个用于定义消息长度相关信息的参数。

实际消息结构会因所用的协议而变化。 三个参数如下所示：

- n： 消息中出现长度说明符的字符位置（从 1 开始）
- 长度大小： 长度说明符的字节数（1、2 或 4）
- 长度 m： 跟在长度说明符后、不包括在长度计数范围内的字符数

这些区域位于设备属性的接收消息组态中。

实例 1： 假设某条消息是根据以下协议构造的：

STX	Len (n)	长度计数包括字符 3 到 14												
		ADR	PKE		INDEX		PWD		STW		HSW		BCC	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	
STX	0x0C	xx	xxxx		xxxx		xxxx		xxxx		xxxx		xx	

请按以下说明组态该消息的接收消息长度参数：

- n = 2（消息长度从字节 2 开始。）
- 长度大小 = 1（消息长度在一个字节中定义。）
- 长度 m = 0（长度说明符后没有不包括在长度计数中的字符。 长度说明符后有 12 个字符。）

在本例中，从 3 到 14（包括 3 和 14）的字符都是 Len (n) 计数的字符。

实例 2： 假设另一条消息是根据以下协议构造的：

SD1	Len (n)	Len (n)	SD2	长度计数包括字符 5 到 10						FCS	ED
				DA	SA	FA	数据单元 = 3 个字节				
1	2	3	4	5	6	7	8	9	10	11	12
xx	0x06	0x06	xx	xx	xx	xx	xx	xx	xx	xx	xx

请按以下说明组态该消息的接收消息长度参数：

- n = 3（消息长度从字节 3 开始。）
- 长度大小 = 1（消息长度在一个字节中定义。）
- 长度 m = 3（长度说明符后有 3 个字符不包括在长度计数中。 在本实例的协议中，字符 SD2、FCS 和 ED 不包括在长度计数中。 其它 6 个字符均包括在长度计数中；因此，长度说明符后总共有 9 个字符。）

在本例中，从 5 到 10（包括 3 和 14）的字符都是 Len (n) 计数的字符。

8.5 设计 PtP 通信

STEP 7 Basic 提供了一些扩展指令，使得用户程序能够使用程序中设计和指定的协议来执行点对点通信。 这些指令可以分为以下两类：

- 组态指令
- 通信指令

组态指令

必须先组态通信接口端口以及用于发送数据和接收数据的参数，然后才能通过用户程序执行 PtP 通信。

可以通过设备配置或用户程序中的如下指令，对各个通信模块执行端口组态和消息组态：

- PORT_CFG
- SEND_CFG
- RCV_CFG

通信指令

PtP 通信指令使用户程序能够与通信模块交换消息。 有关使用这些指令传送数据的信息，请参阅数据一致性 (页 94)部分。

所有 PtP 功能都是异步运行的。 用户程序可以使用轮询架构来确定传送和接收的状态。 SEND_PTP 和 RCV_PTP 可以同时执行。 通信模块根据需要对传送和接收消息进行缓冲，最大缓冲区大小为 1024 字节。

通信模块与实际的点对点设备交换消息。 消息协议位于一个缓冲区中，该缓冲区与特定通信端口交换信息。

- SEND_PTP
- RCV_PTP

其它指令可用于复位接收缓冲区，以及获取和设置特定的 RS232 信号。

- RCV_RST
- SGN_GET
- SGN_SET

8.5.1 轮询架构

必须循环/周期性调用 S7-1200 点对点指令以检查收到的消息。 发送轮训可在发送结束时刻即告知用户程序。

轮询架构： 主站

主站的典型轮询顺序如下：

1. SEND_PTP 指令启动到通信模块的传送。
2. 后续扫描期间会执行 SEND_PTP 指令以轮询传送完成状态。
3. 当 SEND_PTP 指令指示传送完成时，用户代码可以准备接收响应。
4. RCV_PTP 指令反复执行以检查响应。 在 CM 收到响应消息后，RCV_PTP 指令将响应复制到 CPU 并指示已接收到新数据。
5. 用户程序随即可处理响应。
6. 转到第 1 步并重复该循环。

轮询架构： 从站

从站的典型轮询顺序如下：

1. 每次扫描用户程序都会执行 RCV_PTP 指令。
2. CM 收到请求后，RCV_PTP 指令将指示新数据准备就绪并将请求复制到 CPU 中。
3. 用户程序随即处理请求并生成响应。
4. 使用 SEND_PTP 指令将该响应往回发送给主站。
5. 反复执行 SEND_PTP 以确保执行传送。
6. 转到第 1 步并重复该循环。

从站在等待响应期间，必须尽量频繁地调用 RCV_PTP，以便能够在主站超时之前接到来自主站的传送。 要完成该任务，用户程序可以从循环 OB 调用 RCV_PTP，且循环时间应足够大，以便能在超时时间用完之前接到来自主站的传送。 如果将 OB 循环时间设置为在主站的超时时间内可执行该指令两次，则用户程序就一定会接到主站的传送而不会错过任何传送。

8.6 点对点指令

8.6.1 点对点指令的公共参数

通信模块 LED 的行为

通信模块 (CM) 上有 3 个 LED 指示灯：

- 诊断 LED： 在 CPU 找到通信模块前，诊断 LED 将一直以红色闪烁。 CPU 在上电后将检查模块并对 CM 模块进行寻址。 诊断 LED 开始以绿色闪烁。 也就是说，CPU 找到了 CM，但尚未提供该 CM 的组态。该组态在将程序下载到 CPU 时下载到模块。 执行下载到 CPU 操作后，通信模块上的诊断 LED 应为绿色常亮。
- 发送 LED： 发送 LED 在接收 LED 的上方。 从通信端口向外传送数据时，发送 LED 将点亮。
- 接收 LED： 通信端口接收数据时，该 LED 将点亮。

位时间精度

有几个参数以位时间（通过组态的波特率确定）为单位指定的。 以位时间为单位指定参数可以使参数与波特率无关。所有以位时间为单位的参数都可以被指定为最大值 65535 个位。但 S7-1200 可以测量的最长时间是 8 秒。

REQ 输入参数

许多点对点 (PtP) 指令都使用 REQ 输入，该输入在从低电平跳变到高电平时会启动操作。 REQ 输入在指令执行一次的时间内必须为高电平 (TRUE)，不过，REQ 输入可以在用户需要的时间内一直保持为 TRUE。 在 REQ 输入为 FALSE 调用指令以便能复位 REQ 输入的历史状态之前，指令不会启动其它操作。 只有这样，指令才能检测低电平到高电平的跳变以启动下一个操作。

放置 PtP 指令时，系统会提示用户指定背景数据块。 每种 PtP 指令都使用一个唯一的背景数据块。 即，用于给定端口的所有 SEND_PTP 指令应具有同一个背景数据块，但 SEND_PTP 和 RCV_PTP 必须具有不同的背景数据块。 这样可确保各指令能够正确处理一些输入（如 REQ）。

PORT 输入参数

从与 PORT 输入关联的下拉菜单中，选择您希望该指令的实例要操作的 CM 的端口标识符。 该编号也会作为“硬件标识符”出现在 CM 的组态信息中。

NDR、DONE、ERROR 和 STATUS 输出参数

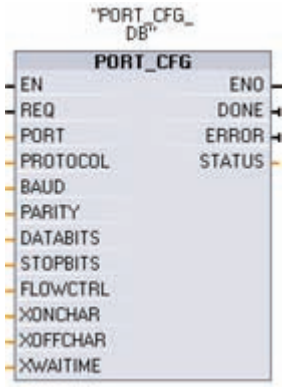
- 输出 DONE 指示所请求的操作已完成且没有错误。 该输出将被置位一个扫描周期时间。
- 输出 NDR（New Data Ready，新数据就绪）指示所请求的操作已完成且没有错误，并且已接收到新数据。 该输出将被置位一个扫描周期时间。
- 输出 ERROR 指示所请求的操作已完成但有错误。 该输出将被置位一个扫描周期时间。
- 输出 STATUS 用于报告错误或中间状态的结果。
  - 如果设置了 DONE 或 NDR 位，则 STATUS 将被设置为 0 或信息代码。
  - 如果 ERROR 位置位，则 STATUS 将被设置为一个错误代码。
  - 如果没有设置以上任何一位，则指令会返回说明功能当前状态的状态结果，例如，忙状态。

公共条件代码

STATUS (W#16#....)	说明
0000	无错误
8x3A	参数 x 中的指针非法
8070	所有内部实例存储器都在使用
8080	端口号非法
8081	超时、模块错误或其它内部错误
8082	由于正在后台进行参数化，参数化失败
8083	缓冲区溢出： CM 返回一条已接收消息，该消息长度大于长度参数所允许的值。
8090	错误的消息长度、错误的子模块或非法消息

STATUS (W#16#....)	说明
8091	参数化消息中的版本错误
8092	参数化消息中的记录长度错误

8.6.2 PORT_CFG 指令



使用 PORT_CFG（端口组态）可以通过用户程序更改端口参数，如波特率等参数。

可以在设备配置属性中设置端口的初始静态组态，或者仅使用默认值。 可以在用户程序中执行 PORT_CFG 指令来更改该组态。 PORT_CFG 组态更改不会永久存储在 CPU 中。 CPU 从 RUN 模式切换到 STOP 模式和循环上电后将恢复设备配置中组态的参数。

更多信息，请参见组态通信端口 (页 274)和管理流控制 (页 275)。

参数	参数类型	数据类型	说明
REQ	IN	Bool	在该输入的上升沿激活组态更改。
PORT	IN	PORT	通信端口标识符： 该逻辑地址是一个可在默认变量表的“常量”(Constants) 选项卡内引用的常量。
PROTOCOL	IN	UInt	0 - 点对点通信协议 1..n - 用于在将来定义特定的协议

参数	参数类型	数据类型	说明
BAUD	IN	UInt	端口波特率： 1 - 300 波特 2 - 600 波特 3 - 1200 波特 4 - 2400 波特 5 - 4800 波特 6 - 9600 波特 7 - 19200 波特 8 - 38400 波特 9 - 57600 波特 10 - 76800 波特 11 - 115200 波特
PARITY	IN	UInt	端口奇偶校验： 1 - 无奇偶校验 2 - 偶校验 3 - 奇校验 4- 传号校验 5 - 空号校验
DATABITS	IN	UInt	每个字符的位数： 1 - 8 个数据位 2 - 7 个数据位
STOPBITS	IN	UInt	停止位： 1 - 1 个停止位 2 - 2 个停止位
FLOWCTRL	IN	UInt	流控制： 1 - 无流控制 2 - XON/XOFF 3 - 硬件 RTS 始终激活 4 - 硬件 RTS 切换
XONCHAR	IN	Char	指定用作 XON 字符的字符。 这通常是 DC1 字符 (11H)。只有启用流控制时，才会评估该参数。
XOFFCHAR	IN	Char	指定用作 XOFF 字符的字符。 这通常是 DC3 字符 (13H)。 只有启用流控制时，才会评估该参数。

参数	参数类型	数据类型	说明
XWAITIME	IN	UInt	指定在接收 XOFF 字符后等待 XON 字符的时间，或者指定在启用 RTC 后等待 CTS 信号的时间（0 到 65535 ms）。只有启用流控制时，才会评估该参数。
DONE	OUT	Bool	上一请求已完成且没有出错后，保持为 TRUE 一个扫描周期时间
ERROR	OUT	Bool	上一请求已完成但出现错误后，保持为 TRUE 一个扫描周期时间
STATUS	OUT	Word	执行条件代码

STATUS (W#16#....)	说明
80A0	特定协议不存在。
80A1	特定波特率不存在。
80A2	特定奇偶校验选项不存在。
80A3	特定数据位数不存在。
80A4	特定停止位数不存在。
80A5	特定流控制类型不存在。
80A6	等待时间为 0 且流控制启用
80A7	XON 和 XOFF 是非法值

8.6.3 SEND_CFG 指令

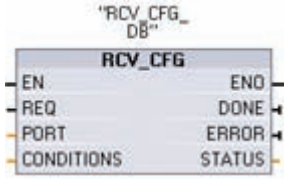
SEND_CFG（发送组态）可用于动态组态点对点通信端口的串行传输参数。 一旦执行 SEND_CFG，便会放弃通信模块 (CM) 内所有排队的消息。

可以在设备配置属性中设置端口的初始静态组态，或者仅使用默认值。 可以在用户程序中执行 SEND_CFG 指令来更改该组态。 SEND_CFG 组态变化不会永久存储在 PLC 中。 CPU 从 RUN 模式切换到 STOP 模式和循环上电后将恢复设备配置中组态的参数。 请参见组态传送（发送）和接收参数 (页 277)。

参数	参数类型	数据类型	说明
REQ	IN	Bool	在该输入的上升沿激活组态更改。
PORT	IN	PORT	通信端口标识符： 该逻辑地址是一个可在默认变量表的“常量”(Constants) 选项卡内引用的常量。
RTSONDLY	IN	UInt	启用 RTS 后执行任何 Tx 数据传输前要等待的毫秒数。 只有启用硬件流控制时，该参数才有效。 0 - 65535 ms。0 将禁用该功能。
RTSOFFDLY	IN	UInt	执行 Tx 数据传输后禁用 RTS 前要等待的毫秒数：只有启用硬件流控制时，该参数才有效。 0 - 65535 ms。0 将禁用该功能。
BREAK	IN	UInt	该参数指定在各消息开始时将发送指定位时间的中断。最大值是 65535 个位的时间。 0 将禁用该功能。 最多 8 秒
IDLELINE	IN	UInt	该参数指定在各消息开始前线路将保持空闲指定的位时间。最大值是 65535 个位的时间。 0 将禁用该功能。 最多 8 秒
DONE	OUT	Bool	上一请求已完成且没有出错后，保持为 TRUE 一个扫描周期时间
ERROR	OUT	Bool	上一请求已完成但出现错误后，保持为 TRUE 一个扫描周期时间
STATUS	OUT	Word	执行条件代码

STATUS (W#16#....)	说明
80B0	不允许传送中断组态
80B1	中断时间大于允许值（2500 个位的时间）
80B2	空闲时间大于允许值（2500 个位的时间）

8.6.4 RCV_CFG 指令



RCV_CFG（接收组态）用于动态组态点对点通信端口的串行接收方参数。 该指令可组态表示接收消息开始和结束的条件。 执行 RCV_CFG 时，将放弃 CM 内所有排队的消息。

可以在设备配置属性中设置 CM 端口的初始静态组态，或者仅使用默认值。 可以在用户程序中执行 RCV_CFG 指令来更改该组态。 RCV_CFG 组态变化不会永久存储在 PLC 中。 CPU 从 RUN 模式切换到 STOP 模式和循环上电后将恢复设备配置中组态的参数。 更多信息，请参见组态接收参数 (页 277)。

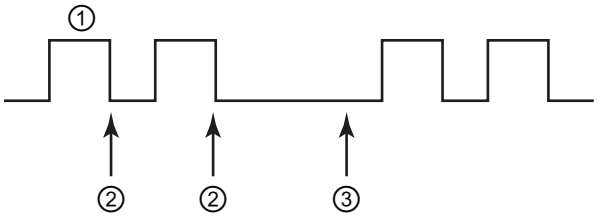
参数	参数类型	数据类型	说明
REQ	IN	Bool	在该输入的上升沿激活组态更改。
PORT	IN	PORT	通信端口标识符： 该逻辑地址是一个可在默认变量表的“常量”(Constants) 选项卡内引用的常量。
CONDITIONS	IN	CONDITIONS	条件数据结构指定消息开始和结束条件。 下文介绍了这些条件。
DONE	OUT	Bool	上一请求已完成且没有出错后，保持为 TRUE 一个扫描周期时间
ERROR	OUT	Bool	上一请求已完成但出现错误后，保持为 TRUE 一个扫描周期时间
STATUS	OUT	Word	执行条件代码

RCV_PTP 指令的开始条件

RCV_PTP 指令使用 RCV_CFG 指令指定的组态来确定点对点通信消息的开始和结束。 消息开始由开始条件确定。 消息开始可以由一个开始条件或开始条件的组合来确定。 如果指定多个开始条件，则只有满足所有条件后才能使消息开始。 可能的开始条件有：

- “开始字符”指定在成功接收到特定字符时开始消息传输。 该字符将是消息中的第一个字符。 在该特定字符前接到的任何字符都将被丢弃。
- “任意字符”指定成功接收的任何字符都将导致消息开始。该字符将是消息中的第一个字符。
- “线路中断”指定应在接收中断字符后开始消息接收操作。

- “线路空闲”指定在接收线路空闲或平静了指定位时间后开始消息接收操作。 一旦出现该条件，就会导致消息开始。



① 字符

② 重新启动线路空闲定时器

③ 检测到线路空闲并启动消息接收操作

- 可变序列： 用户可以构造字符序列数（最多 4 个）可变的开始条件，这些字符序列由数量可变的字符（最多 5 个）组成。 每个序列中的每个字符位置都可以选作特定字符或通配符字符（即任何字符都适合）。 要通过不同字符序列指示消息开始时，可以使用该开始条件。

  请注意以下所接收的十六进制编码消息： “68 10 aa 68 bb 10 aa 16”以及下表中列出的已组态开始序列。 在成功接收到第一个 68H 字符时，开始评估开始序列。 在成功接收到第四个字符（第二个 68H）时，开始条件 1 得到满足。 只要满足了开始条件，就会开始评估结束条件。

  开始序列处理会因各种奇偶校验、成帧或字符间时间错误而终止。 由于不再满足开始条件，因而这些错误将导致不会有接收消息。

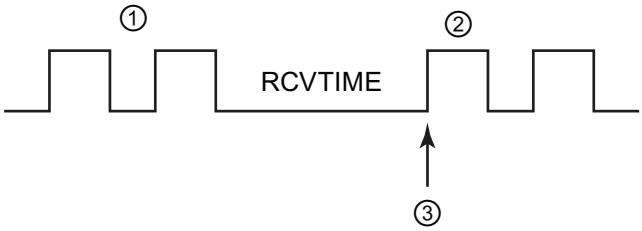
开始条件	第一个字符	第一个字符 +1	第一个字符 +2	第一个字符 +3	第一个字符 +4
1	68H	xx	xx	68H	xx
2	10H	aaH	xx	xx	xx
3	dcH	aaH	xx	xx	xx
4	e5H	xx	xx	xx	xx

RCV_PTP 指令的结束条件

消息结束由指定的结束条件确定。 消息结束由第一次出现的一个或多个已组态结束条件来确定。 可能的消息结束条件有：

- “响应超时”指定应在 RCVTIME 指定的时间内成功接收到的响应字符。 只要传送成功完成且模块开始接收操作，定时器就会启动。 如果在 RCVTIME 时段内没有接收到字

符，将向相应的 RCV_PTP 指令返回错误。 响应超时不定义具体结束条件。 它仅指定应在指定时间内成功接收字符。 必须使用明确的结束条件来定义响应消息的结束条件。

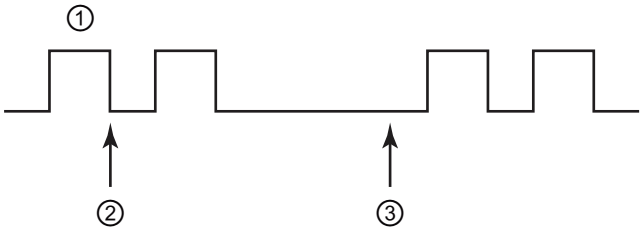


① 传送的字符

② 接收的字符

③ 必须在该时间之前成功接收到第一个字符

- “消息超时”指定应在 MSGTIME 指定的时间内成功接收到消息。 只要满足指定的开始条件，定时器就会启动。

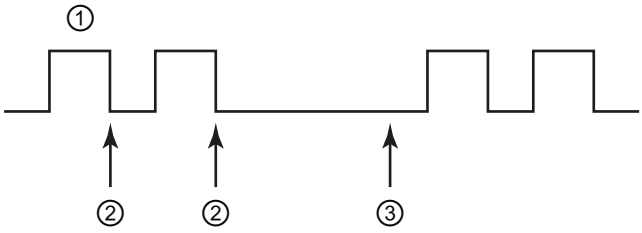


① 接收的字符

② 满足消息开始条件： 消息定时器启动

③ 消息定时器时间已到并终止消息

- 字符间隙是指从一个字符结束（最后一个停止位）到下一个字符结束所测量的时间。如果任何两个字符间的时间超过所组态的位时间数，消息将被终止。



① 接收的字符

② 重新启动字符间定时器

③ 字符间定时器时间用完，同时消息被终止且包含错误

- 最大长度： 接收到指定的字符数后，接收操作停止。 使用该条件可以防止消息缓冲区超负荷运行错误。

  如果将该结束条件与超时结束条件结合使用，在出现超时条件时，即使未达到最大长度也会提供所有有效的已接收字符。 仅当最大长度已知时，该条件才支持长度可变的协议。

- “N + 长度大小 + 长度 M”的组合条件。 该结束条件可用于处理包含长度域且大小可变的消息。

  – N 指定长度域开始的位置（消息中的字符数）。 （从 1 开始）

  – “长度大小”指定长度域的大小。 有效值为 1、2 或 4 个字节。

  – “长度 M”指定不包含在消息长度中的结束字符（跟在长度域后）数。 该值可用于指定大小不包含在长度域中的校验和域的长度。

  – 例如，假设消息由一个开始字符、一个地址字符、一个一字节长度域、消息数据、校验和字符以及一个结束字符组成。 用“Len”表示的条目与 N 参数相对应。 N 的值可以是 3，表示长度字节在消息中的字节 3 中。 “长度大小”的值可以是 1，表示消息长度值包含在 1 个字节中。 校验和与结束字符域与“长度 M”参数相对应。 “长度 M”的值可以是 3，用于指定校验和和字符域的字节数。

开始字符 (1)	地址 (2)	Len (N) (3)	消息 ... (x)		校验和与结束字符 长度 M x+1 x+2 x+3		
xx	xx	xx	xx	xx	xx	xx	xx

- 可变字符： 该结束条件可用于根据不同的字符序列结束接收操作。 这些序列可以由数量可变的字符（最大为 5 个）组成。 每个序列中的每个字符位置都可以选作特定字符或通配符字符（即任何字符都满足条件）。 被组态要忽略的任何前导字符都不要求是消息的一部分。 任何被忽略的尾随字符都要求是消息的一部分。

参数 CONDITIONS 数据类型结构的第 1 部分（开始条件）

参数	参数类型	数据类型	说明
STARTCOND	IN	UInt	指定开始条件： • 01H - 开始字符 • 02H - 任意字符 • 04H - 线路中断 • 08H - 线路空闲 • 10H - 序列 1 • 20H - 序列 2 • 40H - 序列 3 • 80H - 序列 4
IDLETIME	IN	UInt	线路空闲超时所需的位时间数。 仅与线路空闲条件一起使用。0 到 65535
STARTCHAR	IN	Byte	用于开始字符条件的开始字符。
STRSEQ1CTL	IN	Byte	针对每个字符执行的序列 1 忽略/比较控制： 它们是为开始序列中各字符启用的位。 • 01H - 字符 1 • 02H - 字符 2 • 04H - 字符 3 • 08H - 字符 4 • 10H - 字符 5 禁用与某个字符关联的位意味着该序列位置中的任意字符均符合条件。
STRSEQ1	IN	Char[5]	序列 1 开始字符（5 个字符）
STRSEQ2CTL	IN	Byte	针对每个字符执行的序列 2 忽略/比较控制
STRSEQ2	IN	Char[5]	序列 2 开始字符（5 个字符）
STRSEQ3CTL	IN	Byte	针对每个字符执行的序列 3 忽略/比较控制
STRSEQ3	IN	Char[5]	序列 3 开始字符（5 个字符）
STRSEQ4CTL	IN	Byte	针对每个字符执行的序列 4 忽略/比较控制
STRSEQ4	IN	Char[5]	序列 4 开始字符（5 个字符）

参数 CONDITIONS 数据类型结构的第 2 部分（结束条件）

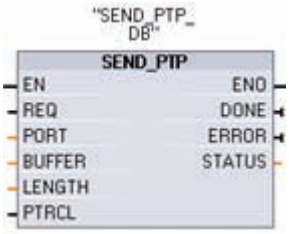
参数	参数类型	数据类型	说明
ENDCOND	IN	UInt	该参数指定消息结束条件： • 01H - 响应时间 • 02H - 消息时间 • 04H - 字符间隙 • 08H - 最大长度 • 10H - N + LEN + M • 20H - 序列
MAXLEN	IN	UInt	最大消息长度： 仅当选择最大长度结束条件时使用。0 到 1023 个字节
N	IN	UInt	长度域在消息中的字节位置。 仅与 N + LEN + M 结束条件一起使用。1 到 1023 个字节
LENGTHSIZE	IN	UInt	字节域的大小（1、2 或 4 个字节）。 仅与 N + LEN + M 结束条件一起使用。
LENGTHM	IN	UInt	指定跟在长度域后、不包含在长度域值内的字符数。 该参数仅与 N + LEN + M 结束条件一起使用。0 到 255 个字节
RCVTIME	IN	UInt	指定接收第一个字符所需的等待时间。 如果在指定时间内没有成功接收到字符，接收操作将被终止且包含错误。 该参数仅与响应时间条件一起使用。 0 到 65535 个位时间，最多 8 秒 因为它仅评估开始条件，所以不会真正将该参数作为结束条件进行评估。 必须选择不同的结束条件。
MSGTIME	IN	UInt	指定在接收到第一个字符后完成接收整条消息所需的等待时间。 只有选择了消息超时条件时，才会使用该参数。 0 - 65535 毫秒
CHARGAP	IN	UInt	指定字符间的位时间数。 如果字符间的位时间数超出指定值，则结束条件得到满足。 该参数仅与字符间隙条件一起使用。 0 到 65535 毫秒

参数	参数类型	数据类型	说明
ENDSEQ1CTL	IN	Byte	针对每个字符执行的序列 1 忽略/比较控制： 它们是为结束序列中各字符启用的位。 字符 1 是位 0，字符 2 是位 1，依此类推，字符 5 是位 4。禁用与某个字符关联的位意味着该序列位置中的任意字符均符合条件。
ENDSEQ1	IN	Char[5]	序列 1 开始字符（5 个字符）

条件代码

STATUS (W#16#....)	说明
80C0	所选开始条件非法
80C1	所选结束条件非法；未选择结束条件
80C2	启用了接收中断，但不允许此操作
80C3	启用了最大长度结束条件，但最大长度是 0 或大于 1024
80C4	启用了计算长度，但 N >= 1023
80C5	启用了计算长度，但长度不是 1、2 或 4
80C6	启用了计算长度，但 M 值大于 255
80C7	启用了计算长度，但计算长度大于 1024
80C8	启用了响应超时，但响应超时为零
80C9	启用了字符间隙超时，但该字符间隙超时为 0 或大于 2500
80CA	启用了线路空闲超时，但该线路空闲超时为 0 或大于 2500
80CB	启用了结束序列，但所有字符均“不相关”
80CC	启用了开始序列（4 个中的任何一个），但所有字符均“不相关”

8.6.5 SEND_PTP 指令



SEND_PTP（发送点对点数据）用于启动数据传送。SEND_PTP 将指定的缓冲区数据传送到 CM。在 CM 以指定波特率发送数据的同时，CPU 程序会继续执行。 仅一个发送操作可以在某一给定时间处于未决状态。 如果在 CM 已经开始传送消息时执行第二个 SEND_PTP，CM 将返回错误。

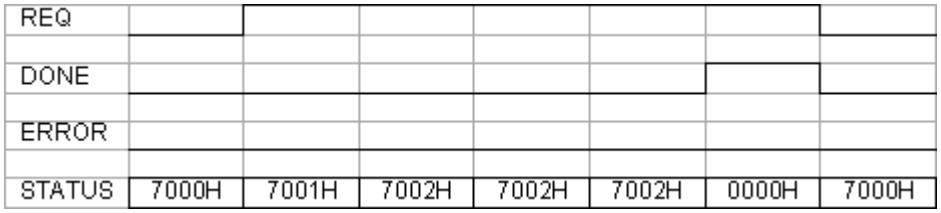
参数	参数类型	数据类型	说明
REQ	IN	Bool	在该传送使能输入的上升沿激活所请求的传送。这会启动将缓冲区数据传送到点对点通信模块 (CM)。
PORT	IN	PORT	通信端口标识符： 该逻辑地址是一个可在默认变量表的“常量”(Constants) 选项卡内引用的常量。
BUFFER	IN	Variant	该参数指向传送缓冲区的起始位置。 不支持布尔数据或布尔数组。
LENGTH	IN	UInt	用字节表示的传输的消息帧长度 传输复杂结构时，始终使用长度 0。
PTRCL	IN	Bool	该参数选择普通点对点协议或 Siemens 提供的特定协议所在的缓冲区，这些协议在所连接的 CM 中实施。 FALSE = 用户程序控制的点对点操作。 （仅限有效选项）
DONE	OUT	Bool	上一请求已完成且没有出错后，保持为 TRUE 一个扫描周期时间
ERROR	OUT	Bool	上一请求已完成但出现错误后，保持为 TRUE 一个扫描周期时间
STATUS	OUT	Word	执行条件代码

传送操作进行期间，DONE 和 ERROR 输出均为 FALSE。 传送操作完成后，DONE 或 ERROR 输出将被设置为 TRUE（持续一个扫描周期）以显示传送操作的状态。 当 DONE 或 ERROR 为 TRUE 时，STATUS 输出有效。

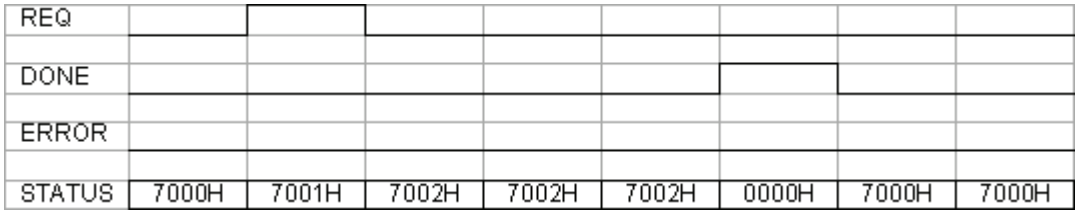
如果通信模块 (CM) 接受所传送的数据，则该指令将返回状态 16#7001。 如果 CM 仍在忙于传送，则后续的 SEND_PTP 执行将返回 16#7002。 传送操作完成后，如果未出错，CM 将返回传送操作状态 16#0000。 后续执行 REQ 为低电平的 SEND_PTP 时，将返回状态 16#7000（不忙）。

输出值与 REQ 的关系如下：

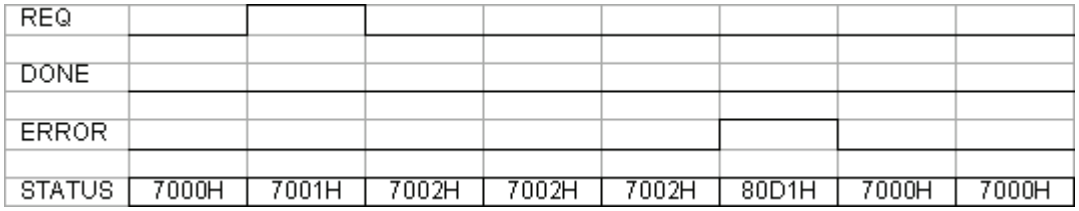
假设定期调用该指令以检查传送过程的状态。 在下图中，假设每次扫描都调用该指令（用 STATUS 值表示）。



下图显示通过 REQ 线路脉冲（持续一个扫描周期）启动传送操作时，DONE 和 STATUS 参数是如何仅在一个扫描周期内有效。



下图显示了出错时 DONE、ERROR 和 STATUS 参数之间的关系。



STATUS (W#16#....)	说明
80D0	传送方激活期间发出新请求
80D1	由于在等待时间内没有 CTS 信号，传送中止
80D2	由于没有来自 DCE 设备的 DSR，传送中止
80D3	由于队列溢出（传送 1024 个字节以上），传送中止
7000	不忙

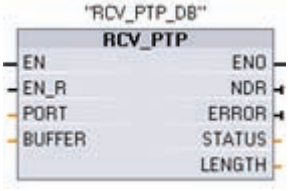
STATUS (W#16#....)	说明
7001	接受请求时正忙（第一次调用）
7002	轮询时正忙（第 n 次调用）

PTP_SEND 的 LENGTH 和 DATA 参数的交互作用

PTP_SEND 指令可以传送的最小数据单位是字节。 DATA 参数决定要发送的数据的大小。 对于 DATA 参数，无法使用 BOOL 和 BOOL 数组。

LENGTH 参数	DATA 参数	说明
LENGTH = 0	未使用	发送在 DATA 参数中定义的全部数据。 当 LENGTH = 0 时，用户无须指定发送字节数。
LENGTH > 0	基本数据类型	LENGTH 值必须包含此数据类型的字节计数。 否则，不会传送任何数据并返回错误 8088H。
	结构	LENGTH 值可以包含小于结构完整字节长度的字节数。 在这种情况下，仅传送前 LENGTH 个字节。
	数组	LENGTH 值可以包含小于数组完整字节长度的字节数。 在这种情况下，仅传送完全适合 LENGTH 个字节的数组元素。 LENGTH 值必须为数据元素字节数的倍数。 否则，STATUS = 8088H、ERROR = 1 且不进行任何传送。
	字符串	传送整个字符串格式的存储区数据。 LENGTH 值必须包含用于最大长度、实际长度和字符串字符的字节数。 对于 STRING 数据类型，所有长度和字符为一个字节大小。 如果将一个字符串用作 DATA 参数的实际参数，则 LENGTH 值还必须包含两个字节来指示两个长度域。

8.6.6 RCV_PTP 指令



RCV_PTP（接收点对点）检查 CM 中已接收的消息。如果有消息，则会将其从 CM 传送到 CPU。 如果发生错误，则会返回相应的 STATUS 值。

NDR 或 ERROR 为 TRUE 时，STATUS 值有效。 STATUS 值提供 CM 中的接收操作终止的原因。它通常是正值，表示接收操作成功且接收过程正常终止。 如果 STATUS 值为负数（十六进制值的最高有效位置位），则表示接收操作因错误条件终止，例如，奇偶校验、组帧或超限错误。

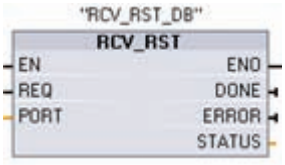
每个点对点 CM 模块最多可以缓冲最大值 1K 字节。 这可以是一个大消息或几个较小的消息。

参数	参数类型	数据类型	说明
EN_R	IN	Bool	该输入为 TRUE 时，检查 CM 模块是否已接收消息。 如果已成功接收消息，则会将其从模块传送到 CPU。 EN_R 为 FALSE 时，将检查 CM 是否收到消息并设置 STATUS 输出，但不会将消息传送到 CPU。
PORT	IN	PORT	通信端口标识符： 该逻辑地址是一个可在默认变量表的“常量”(Constants) 选项卡内引用的常量。
BUFFER	IN	Variant	该参数指向接收缓冲区的起始位置。 该缓冲区应该足够大，可以接收最大长度消息。 不支持布尔数据或布尔数组。
NDR	OUT	Bool	新数据就绪且操作无错误地完成时，在一个扫描周期内为 TRUE。
ERROR	OUT	Bool	操作已完成但出现错误，在一个扫描周期内为 TRUE
STATUS	OUT	Word	执行条件代码
LENGTH	OUT	UInt	返回消息的长度（字节）

STATUS (W#16#...)	说明
0000	没有提供缓冲区
80E0	因接收缓冲区已满，消息被终止
80E1	因出现奇偶校验错误，消息被终止
80E2	因组帧错误，消息被终止
80E3	因出现超限错误，消息被终止

STATUS (W#16#...)	说明
80E4	因计算长度超出缓冲区大小，消息被终止
0094	因接收到最大字符长度，消息被终止
0095	因消息超时，消息被终止
0096	消息因字符间超时而终止
0097	消息因响应超时而终止
0098	因已满足“N+LEN+M”长度条件，消息被终止
0099	因已满足结束序列，消息被终止

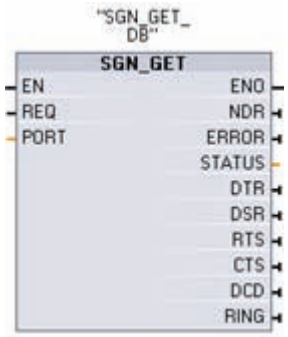
8.6.7 RCV_RST 指令



RCV_RST（接收方复位）可清空 CM 中的接收缓冲区。

参数	参数类型	数据类型	说明
REQ	IN	Bool	该使能输入的上升沿激活接收方复位
PORT	IN	PORT	通信端口标识符： 端口必须使用模块的逻辑地址指定。
DONE	OUT	Bool	在一个扫描周期内为 TRUE 时，表示上一个请求已完成且没有错误。
ERROR	OUT	Bool	为 TRUE 时，表示上一个请求已完成但有错误。 此外，该输出为 TRUE 时，STATUS 输出还会包含相关错误代码。
STATUS	OUT	Word	错误代码

8.6.8 SGN_GET 指令



SGN_GET（获取 RS232 信号）读取 RS232 通信信号的当前状态。 该功能仅对 RS232 CM（通信模块）有效。

参数	参数类型	数据类型	说明
REQ	IN	Bool	在该输入的上升沿获取 RS232 信号状态值
PORT	IN	PORT	通信端口标识符： 该逻辑地址是一个可在默认变量表的“常量”(Constants) 选项卡内引用的常量。
NDR	OUT	Bool	新数据就绪且操作无错误地完成时，在一个扫描周期内为 TRUE
ERROR	OUT	Bool	操作已完成但出现错误，在一个扫描周期内为 TRUE
STATUS	OUT	Word	执行条件代码
DTR	OUT	Bool	数据终端就绪，模块就绪（输出）
DSR	OUT	Bool	数据设备就绪，通信伙伴就绪（输入）
RTS	OUT	Bool	请求发送，模块已做好发送准备（输出）
CTS	OUT	Bool	允许发送，通信伙伴可以接收数据（输入）
DCD	OUT	Bool	数据载波检测，接收信号电平（始终为假，不支持）
RING	OUT	Bool	响铃指示器，来电指示（始终为假，不支持）

STATUS (W#16#....)	说明
80F0	CM 是 RS485 模块且没有信号可用
80F1	信号因硬件流控制而无法设置

STATUS (W#16#....)	说明
80F2	因模块是 DTE 而无法设置 DSR
80F3	因模块是 DCE 而无法设置 DTR

8.6.9 SGN_SET 指令

SGN_SET（设置 RS232 信号）设置 RS232 通信信号的状态。该功能仅对 RS232 CM（通信模块）有效。

参数	参数类型	数据类型	说明
REQ	IN	Bool	在该输入的上升沿启动设置 RS232 信号的操作
PORT	IN	PORT	通信端口标识符： 该逻辑地址是一个可在默认变量表的“常量”(Constants) 选项卡内引用的常量。
SIGNAL	IN	Byte	选择要设置的信号：（允许多个） • 01H = 设置 RTS • 02H = 设置 DTR • 04H = 设置 DSR
RTS	IN	Bool	请求发送，模块准备好将值发送到设备（真或假）
DTR	IN	Bool	数据终端就绪，模块准备好将值发送到设备（真或假）
DSR	IN	Bool	数据设备就绪（仅适用于 DCE 型接口）（不使用）
DONE	OUT	Bool	上一请求已完成且没有出错后，保持为 TRUE 一个扫描周期时间
ERROR	OUT	Bool	上一请求已完成但出现错误后，保持为 TRUE 一个扫描周期时间
STATUS	OUT	Word	执行条件代码

STATUS (W#16#....)	说明
80F0	CM 是 RS485 模块且没有可设置的信号
80F1	信号因硬件流控制而无法设置
80F2	因模块是 DTE 而无法设置 DSR
80F3	因模块是 DCE 而无法设置 DTR

8.7 错误

PtP 指令的返回值

每个 PtP 指令都具有可提供完成状态的三个输出：

参数	数据类型	默认值	说明
DONE	Boolean	FALSE	在一个扫描周期内为 TRUE，表示上一个请求已完成且没有错误。
ERROR	Boolean	FALSE	TRUE 表示上一个请求已完成但有错误，并且 STATUS 中有相应的错误代码。
STATUS	Word	0	包含错误类别和错误编号的两个字节（如果适用）。 STATUS 在该功能执行期间一直保持其值。

常见错误类别和错误

类别说明	错误类别	说明
端口组态	80Ax	用于定义常见端口组态错误
传送组态	80Bx	用于定义常见传送组态错误
接收组态	80Cx	用于定义常见接收组态错误
传送运行时	80Dx	用于定义常见传送运行时错误
接收运行时	80Ex	用于定义常见接收运行时错误
信号处理	80Fx	用于定义与所有信号处理相关的常见错误

端口组态错误

事件/错误 ID	说明
0x80A0	特定协议不存在
0x80A1	特定波特率不存在
0x80A2	特定奇偶校验不存在
0x80A3	特定数据位数不存在
0x80A4	特定停止位数不存在
0x80A5	特定流控制类型不存在

传送组态错误

事件/错误 ID	说明
0x80B0	特定协议不存在
0x80B1	特定波特率不存在
0x80B2	特定奇偶校验不存在
0x80B3	特定数据位数不存在
0x80B4	特定停止位数不存在
0x80B5	特定流控制类型不存在

接收组态错误

事件/错误 ID	说明
0x80C0	开始条件错误
0x80C1	结束条件错误
0x80C3	最大长度错误
0x80C4	N 值错误（请参见 N+LEN+M）
0x80C5	长度大小错误（请参见 MAXLEN 或 N+LEN+M）
0x80C6	M 值错误（请参见 N+LEN+M）
0x80C7	N-长度-M 值错误（请参见 N+LEN+M）

事件/错误 ID	说明
0x80C8	响应超时错误，指定的接收时间段内未收到消息。（请参见 RCVTIME 或 MSGTIME）
0x80C9	字符间超时错误（请参见 CHARGAP）
0x80CA	空闲线路超时错误（请参见空闲线路）
0x80CB	组态了指定的结束序列，但所有字符均“不相关”
0x80CC	组态了指定的开始序列，但所有字符均“不相关”

信号错误

事件/错误 ID	说明
0x80F0	通信模块是 RS485 模块且没有信号可用
0x80F1	通信模块是 RS232 模块，但由于启用了 H/W 流控制而没有可设置的信号
0x80F2	由于模块是 DTE 设备，而无法设置 DSR 信号

传送运行时错误

事件/错误 ID	说明
缓冲区限制	已超出 CP 总的可用传送缓冲区
0x80D0	传送方处于激活状态时收到新请求
0x80D1	接收方发出了暂停主动传输的流控制请求并且在指定的等待时间内未重新激活该传输 在硬件流控制期间，如果接收方在指定的等待时间内没有声明 CTS，也会产生该错误
0x80D2	传送请求中止，因为没有从 DCE 收到任何 DSR 信号
0x80D3	已超出 CP 总的可用传送缓冲区
0x7000	传送功能不忙
0x7001	传送功能忙于处理第一个调用
0x7002	传送功能忙于处理后续调用（第一个调用后的轮询）

接收运行时返回值

事件/错误 ID	说明
0x80E0	因接收缓冲区已满，消息被终止
0x80E1	因出现奇偶校验错误，消息被终止
0x80E2	因组帧错误，消息被终止
0x80E3	因出现超限错误，消息被终止
0x80E4	因指定长度超出总缓冲区大小，消息被终止
0x0094	因接收到最大字符长度 (MAXLEN)，消息被终止
0x0095	因指定时间 (MSGTIME) 内未收到完整消息，消息被终止
0x0096	因在字符间时间 (CHARGAP) 内未收到下一个字符，消息被终止
0x0097	因指定时间 (RCVTIME) 内未收到第一个字符，消息被终止
0x0098	因已满足“n+len+m”长度条件 (N+LEN+M)，消息被终止
0x0099	因已满足结束序列 (ENDSEQ)，消息被终止

其它参数错误

事件/错误 ID	说明
0x8n3A	在参数 n 中提供了非法指针
0x8070	所有内部实例存储器都在使用
0x8080	此端口号无效
0x8082	由于已经在后台进行参数化，参数化失败
0x8083	缓冲区溢出。 CM 返回的数据超出允许的数据量。
0x8085	LEN 参数的值为 0 或比最大的允许值大
0x8088	LEN 参数大于在 DATA 中指定的存储区

9 在线和诊断工具

9.1 状态 LED

CPU 和 I/O 模块使用 LED 提供有关模块或 I/O 的运行状态的信息。 CPU 提供以下状态指示灯：

- STOP/RUN
  - 纯橙色指示 STOP 模式
  - 纯绿色指示 RUN 模式
  - 闪烁（绿色和橙色交替）指示 CPU 正在启动
- ERROR
  - 红色闪烁指示有错误，例如，CPU 内部错误，存储卡错误或组态错误（模块不匹配）
  - 纯红色指示硬件出现故障
- MAINT（维护）在每次插入存储卡时闪烁。 然后 CPU 切换到 STOP 模式。 在 CPU 切换到 STOP 模式后，执行以下操作之一以启动存储卡评估：
  - 将 CPU 切换到 RUN 模式
  - 执行存储器复位 (MRES)
  - CPU 循环上电

说明	STOP/RUN 橙色/绿色	ERROR 红色	MAINT 橙色
断电	灭	灭	灭
启动、自检、固件更新	闪烁 （绿色和橙色交替）	-	灭
停止模式	亮 （橙色）	-	-
运行模式	亮 （橙色）	-	-
取出存储卡	亮 （橙色）	-	闪烁

说明	STOP/RUN 橙色/绿色	ERROR 红色	MAINT 橙色
出错	亮 （橙色或绿色）	闪烁	-
请求维护	亮 （橙色或绿色）	-	亮
硬件出现故障	亮 （橙色）	亮	灭
LED 测试或 CPU 固件出现故障	闪烁 （绿色和橙色交替）	闪烁	闪烁

CPU 还提供了两个可指示 PROFINET 通信状态的 LED。 打开底部端子块的盖子可以看到 PROFINET LED。

- Link （绿色）点亮指示连接成功
- Rx/Tx （黄色）点亮指示传输活动

CPU 和各数字量信号模块 (SM) 为每个数字量输入和输出提供了 I/O Channel LED。 I/O Channel （绿色）通过点亮或熄灭来指示各输入或输出的状态。

此外，各数字量 SM 还提供了指示模块状态的 DIAG LED：

- 绿色指示模块处于运行状态
- 红色指示模块有故障或处于非运行状态

各模拟量 SM 为各路模拟量输入和输出提供了 I/O Channel LED。

- 绿色指示通道已组态且处于激活状态
- 红色指示个别模拟量输入或输出处于错误状态

此外，各模拟量 SM 还提供有指示模块状态的 DIAG LED：

- 绿色指示模块处于运行状态
- 红色指示模块有故障或处于非运行状态

SM 可检测模块的通断电情况（必要时，还可检测现场侧电源）。

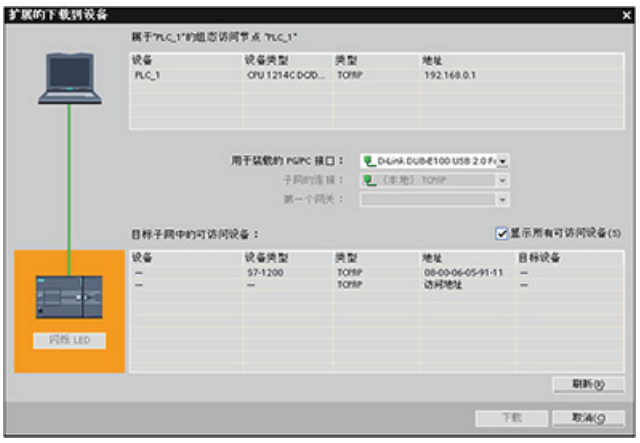
说明	DIAG （红色/绿色）	I/O Channel （红色/绿色）
现场侧电源关闭	呈红色闪烁	呈红色闪烁
没有组态或更新在进行中	呈绿色闪烁	灭
模块已组态且没有错误	亮（绿色）	亮（绿色）
错误状态	呈红色闪烁	-
I/O 错误（启用诊断时）	-	呈红色闪烁
I/O 错误（禁用诊断时）	-	亮（绿色）

9.2 转到在线并连接到 CPU

将程序和项目工程数据加载到目标系统以及执行下列操作时，编程设备和目标系统之间必须存在在线连接：

- 测试用户程序
- 显示和改变 CPU 的工作模式
- 显示和设置 CPU 的日期和日时钟
- 显示模块信息
- 比较在线块和离线块
- 诊断硬件

然后可以在在线或诊断视图中使用“在线工具”(Online tools) 任务卡访问目标系统中的数据。



设备的当前在线状态由项目导航中该设备旁边的图标指示。

橙色指示存在在线连接。

选择“可访问节点”(Accessible Nodes) 可查找网络上的 CPU。

转到在线 单击“转到在线”(Go online) 可连接到网络上的 CPU。

9.3 设置 IP 地址和日时钟

可以设置在线 CPU 中的 IP 地址和日时钟。

从“在线和诊断”(Online & diagnostics) 区域连接到在线 CPU 后，可以显示或更改 IP 地址。

更多信息，请参见 IP 地址 (页 82)部分。

还可以显示或设置在线 CPU 的时间和日期参数。

9.4 在线 CPU 的 CPU 操作员面板

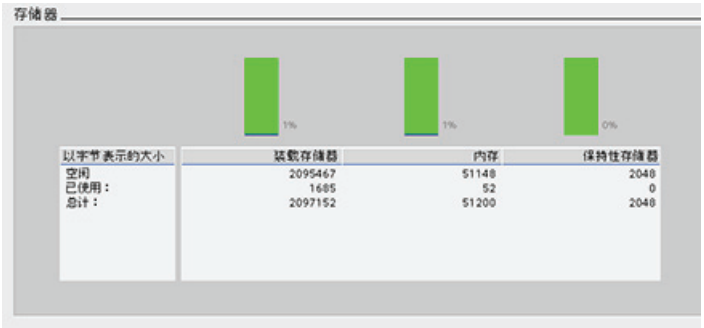
“CPU 操作员面板”(CPU operator panel) 任务卡显示在线 CPU 的工作模式（STOP 或 RUN）： 该面板还显示 CPU 是否有错误或值是否处于强制状态。 CPU 操作面板用于更改在线 CPU 的工作模式。

9.5 监视循环时间和存储器使用情况

可以监视在线 CPU 的循环时间和存储器使用情况。

连接到在线 CPU 后，可以查看以下测量值：

- 循环时间
- 存储器使用情况



9.6 显示 CPU 中的诊断事件

使用诊断缓冲区可以查看 CPU 的近期活动。 诊断缓冲区包含下列条目：

- 诊断事件
- CPU 工作模式改变（切换到 STOP 或 RUN 模式）

第一个条目包含最新的事件。 诊断缓冲区中的各条目均包含记录事件的日期和时间以及一段说明。

最大条目数由 CPU 决定。 最多支持 50 个条目。

仅永久存储诊断缓冲区中 10 个最新的事件。 将 CPU 复位为工厂设置会通过删除条目的方式复位诊断缓冲区。

9.7 用于监视用户程序的监视表格

通过监视表格可以在 CPU 执行用户程序时对数据点执行监视和控制功能。 根据监视或控制功能的不同，这些数据点可以是过程映像（I 或 Q）、物理映像（I_:P 或 Q_:P）、M 或 DB。

监视功能不会改变程序顺序。 它为用户提供有关程序顺序的信息以及 CPU 中的程序的数据。

控制功能允许用户控制程序的顺序和数据。 使用控制功能时必须小心谨慎。 这些功能可能会严重影响用户/系统程序的执行。 三种控制功能是修改、强制和在 STOP 模式下启用输出。

使用监视表格可以执行以下在线功能：

- 监视变量的状态
- 修改个别变量的值
- 将变量强制设置为特定值

选择监视或修改变量的时间：

- 扫描循环开始时： 在该扫描循环开始时读取或写入值
- 扫描循环结束时： 在该扫描循环结束时读取或写入值
- 切换到停止

要创建监视表格：

1. 双击“添加新监视表格”(Add new watch table) 打开新监视表格。
2. 输入变量名称将变量添加到监视表格。

可使用以下选项监视变量：

- “监视全部”(Monitor all)： 该命令用于启动对激活的监视表格中的可见变量进行监视。
- “立即监视”(Monitor now)： 该命令用于启动对激活的监视表格中的可见变量进行监视。 监视表格仅立即监视变量一次。

可使用以下选项修改变量：

- “修改为 0”(Modify to 0) 将所选地址的值设置为“0”。
- “修改为 1”(Modify to 1) 将所选地址的值设置为“1”。
- “立即修改”(Modify now) 立即修改所选地址的值一个扫描周期。
- “使用触发器修改”(Modify with trigger) 修改所选地址的值。

  该功能不提供反馈来指示实际上是否修改了所选地址。 如果需要修改反馈，则使用“立即修改”(Modify now) 功能。
- “启用外围设备输出”(Enable peripheral outputs) 禁用输出禁用命令并且仅在 CPU 处于 STOP 模式时可用。

要监视变量，必须在线连接到 CPU。

		名称	地址	显示格式	监视值	使用触发器监视	使用触发器修改
1		"Start"	%I0.0	布尔型		永久	永久
2		"Stop"	%I0.1	布尔型		永久	永久
3		"Running"	%I0.2	布尔型		永久	永久
4		"Del"	%I0.3	布尔型		永久	永久

可以使用监视表格顶部的按钮选择各种功能。

输入要监视的变量名称并从该下拉选择项中选择一种显示格式。 在线连接到 CPU 时，单击“监视”(Monitor) 按钮将在“监视值”(Monitor value) 域中显示数据点的实际值。

监视或修改 PLC 变量时使用触发器

触发决定将在扫描周期中的哪个点监视或修改所选地址。

触发类型	说明
永久	连续采集数据
扫描周期开始时	永久： CPU 读取输入后，在扫描周期开始时连续采集数据
	一次： CPU 读取输入后，在扫描周期开始时采集一次数据
扫描周期结束时	永久： CPU 写入输出前，在扫描周期结束时连续采集数据
	一次： CPU 写入输出前，在扫描周期结束时采集一次数据
切换到 STOP 时	永久： CPU 切换到 STOP 时连续采集数据
	一次： CPU 切换到 STOP 后采集一次数据

要在给定触发点修改 PLC 变量，请选择周期开始或结束。

- 修改输出： 触发修改输出事件的最佳时机是在扫描周期结束且 CPU 马上要写入输出之前的时间。

  在扫描周期开始时监视输出的值以确定写入到物理输出中的值。 此外，在 CPU 将值写入到物理输出前监视输出以检查程序逻辑并与实际 I/O 行为进行比较。

- 修改输入： 触发修改输入事件的最佳时机是在周期开始、CPU 刚读取输入且用户程序要使用输入值之前的时间。

  如果在扫描周期开始时修改输入，则还应在扫描周期结束时监视输入值，以确保扫描周期结束时的输入值自扫描周期开始起未改变。 如果值不同，则用户程序可能会错误地写入到输入。

要诊断 CPU 转到 STOP 的可能原因，请使用“切换到 STOP”(Transition to STOP) 触发器捕捉上一个过程值。

在 STOP 模式下启用输出

监视表格允许用户在 CPU 处于 STOP 模式时写入输出。 通过该功能可以检查输出的接线并检验连接到输出引脚的电线是将高电平信号还是低电平信号引入与其相连的过程设备端子。

⚠警告
即使在 CPU 处于 STOP 模式时，启用物理输出也可激活相连的过程点。

输出启用时，可以在 STOP 模式下修改输出的状态。 如果输出禁用，则无法在 STOP 模式下修改输出。

- 要启用在 STOP 模式下修改输出，请选择“在线”(Online) 菜单中的“修改”(Modify) 命令的“启用外围设备输出”(Enable peripheral outputs) 选项或右键单击监视表格行。
- 将 CPU 设置为 RUN 模式会禁用“启用外围设备输出”(Enable peripheral outputs) 选项。
- 如果任何输入或输出被强制，则处于 STOP 模式时不允许 CPU 启用输出。 必须先取消强制功能。

CPU 中的强制值

CPU 允许用户通过在监视表格中指定物理输入或输出地址（I_:P 或 Q_:P）并启动强制，以此来强制输入和输出点。

在程序中，物理输入的读取值被强制值覆盖。 程序在处理过程中使用该强制值。 程序写入物理输出时，输出值被强制值覆盖。 强制值出现在物理输出端并被过程使用。

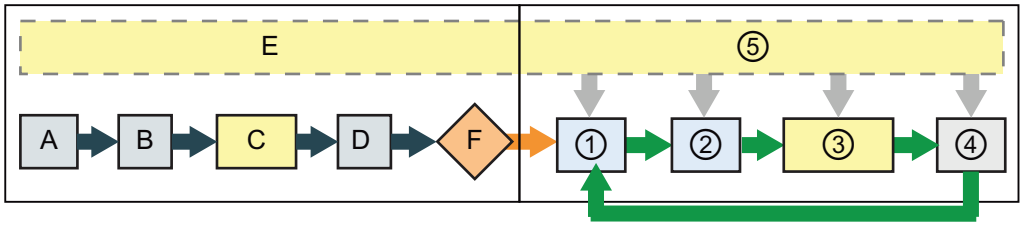
在监视表格中强制输入或输出时，强制操作将变成用户程序的一部分。 即使编程软件已关闭，强制选项在运行的 CPU 程序中仍保持激活，直到在线连接到编程软件并停止强制功能将其清除为止。 含有通过存储卡装载到另一个 CPU 的强制点的程序将继续强制程序中选择的点。

如果 CPU 正在执行写保护存储卡上的用户程序，则无法通过监视表格初始化或更改对 I/O 的强制，因为用户无法改写写保护用户程序中的值。强制写保护值的任何尝试都将生成错误。如果使用存储卡传送用户程序，则该存储卡上的所有被强制元素都将被传送到 CPU。

说明

无法强制分配给 HSC、PWM 和 PTO 的数字 I/O 点

在设备配置期间分配高速计数器 (HSC)、脉冲宽度调制 (PWM) 和脉冲串输出 (PTO) 设备使用的数字 I/O 点。 将数字量 I/O 点的地址分配给这些设备之后，无法通过监视表格的强制功能修改所分配的 I/O 点的地址值。



启动

A 强制功能不影响 I 存储区的清除。

B 强制功能不影响输出值的初始化。

C 启动 OB 执行期间，CPU 在用户程序访问物理输入时应用强制值。

D 不影响将中断事件存储到队列。

E 不影响写入到输出的启用。

RUN

① 将 Q 存储器写入到物理输出时，CPU 在更新输出时应用强制值。

② 读取物理输入时，CPU 仅在将这些输入复制到 I 存储器前应用强制值。

③ 用户程序（程序循环 OB）执行期间，CPU 在用户程序访问物理输入或写入物理输出时应用强制值。

④ 强制功能不影响通信请求和自检诊断的处理。

⑤ 不影响在扫描周期的任何时段内处理中断。

技术规范

A

A.1 常规技术规范

遵守的标准

S7-1200 自动化系统符合以下标准和测试规范。 S7-1200 自动化系统的测试标准均基于这些标准和测试规范。

CE 认证

CE S7-1200 自动化系统满足下列 EC 指令提出的要求和安全相关目标，并且符合欧盟的公报中列出的可编程控制器的协调欧洲标准 (EN)。

- EC 指令 2006/95/EC（低压指令）“设计用于特定电压限值内的电气设备”
  - EN 61131-2:2007 可编程控制器 - 设备要求和测试
- EC 指令 2004/108/EC（EMC 指令）“电磁兼容性”
  - 辐射标准
    EN 61000-6-4:2007： 工业环境
  - 抗扰度标准
    EN 61000-6-2:2005： 工业环境
- EC 指令 94/9/EC (ATEX)“拟用于潜在爆炸性环境的设备和保护系统”
  - EN 60079-15:2005： 保护类型“n”

可向主管部门出具的所持 CE 一致性声明文件位于以下地址：

Siemens AG
IA AS RD ST PLC Amberg
Werner-von-Siemens-Str. 50
D92224 Amberg
Germany

cULus 认证

美国安全检测实验室公司，符合

- 美国安全检测实验室公司： UL 508 认证（工业控制设备）
- 加拿大标准协会： CSA C22.2 第 142 号（过程控制设备）

注意
SIMATIC S7-1200 系列符合 CSA 标准。 cULus 标志表示 S7-1200 已通过美国安全检测实验室公司 (UL) 检验和认证，其符合标准 UL 508 和 CSA 22.2 第 142 号。

FM 认证

工厂共同研究协会 (FM)：
认证标准类别号 3600 和 3611
批准用于：
I 类，2 分区，气体组别 A、B、C、D，温度类别 T4A Ta = 40° C
I 类，2 区，IIC，温度类别 T4 Ta = 40° C

ATEX 认证

EN 60079-0:2006： 爆炸性环境 - 一般要求
EN 60079-15:2005： 用于潜在爆炸性环境的电气装置；保护类型“n”
II 3 G Ex nA II T4

要安全使用 S7-1200，必须遵守以下特殊条件：

- 将模块安装在合适的机柜中，根据 EN 60529 至少要提供防护等级 IP54，并且考虑设备将来使用的环境条件。
- 在额定条件下，如果电缆入口点温度超出 70° C 或者导线分支点超出 80° C，则所选电缆的温度规范应符合实际测量温度。
- 应采取措施防止额定电压受暂态干扰而超出 40% 以上。

C-Tick 认证

S7-1200 自动化系统满足 AS/NZS 2064（A 类）标准的要求

海事认证

S7-1200 产品定期向特定机构递交申请以便进行与特定市场和应用有关的认证。 如需要更多有关按零件号排列的最新具体认证列表的信息，请咨询当地西门子代表。

船级社：

- ABS（American Bureau of Shipping，美国船级社）
- BV（Bureau Veritas，法国船级社）
- DNV（Det Norske Veritas，挪威船级社）
- GL（Germanischer Lloyd，德国船级社）
- LRS（Lloyds Register of Shipping，英国劳氏船级社）
- Class NK（Nippon Kaiji Kyokai，日本船级社）

工业环境

S7-1200 自动化系统设计用在工业环境中。

应用现场	噪声排放要求	噪声抗扰度要求
工业	EN 61000-6-4:2007	EN 61000-6-2:2005

电磁兼容性

电磁兼容性 (EMC) 是电气设备在电磁环境中按预期运行以及运行时电磁干扰的发射水平 (EMI) 不会干扰周围其它电气设备的能力。

电磁兼容性 - 抗扰度符合 EN 61000-6-2	
EN 61000-4-2 静电放电	8 kV，对所有表面的空中放电 6 kV，对暴露导电表面的接触放电
EN 61000-4-3 辐射电磁场	80 到 1000 MHz，10 V/m，1 kHz 时 80% AM 1.4 到 2.0 GHz，3 V/m，1 kHz 时 80% AM 2.0 到 2.7 GHz，1 V/m，1 kHz 时 80% AM
EN 61000-4-4 快速瞬变脉冲	2 kV，5 kHz，到 AC 和 DC 系统电源的耦合网络 2 kV，5 kHz，到 I/O 的耦合夹
EN 6100-4-5 浪涌抗扰度	AC 系统 - 2 kV 共模，1kV 差模 DC 系统 - 2 kV 共模，1kV 差模 对于 DC 系统（I/O 信号、DC 电源系统)，需要外部保护。
EN 61000-4-6 传导干扰	150 kHz 到 80 MHz，10 V RMS，1kHz 时 80% AM
EN 61000-4-11 电压骤降	AC 系统 60 Hz 时，0% 持续 1 个周期、40% 持续 12 个周期和 70% 持续 30 个周期

电磁兼容性 - 传导和辐射发射符合 EN 61000-6-4	
传导发射 EN 55011，A 类，组 1 0.15 MHz 到 0.5 MHz 0.5 MHz 到 5 MHz 5 MHz 到 30 MHz	 <79dB (µV) 准峰值；<66 dB (µV) 平均值 <73dB (µV) 准峰值；<60 dB (µV) 平均值 <73dB (µV) 准峰值；<60 dB (µV) 平均值
辐射发射 EN 55011，A 类，组 1 30 MHz 到 230 MHz 230 MHz 到 1 GHz	 <40dB (µV/m) 准峰值；在 10m 处测得 <47dB (µV/m) 准峰值；在 10m 处测得

环境条件

环境条件 - 运输和存储	
EN 60068-2-2，测试 Bb，干热和 EN 60068-2-1，测试 Ab，寒冷	-40° C 到 +70° C
EN 60068-2-30，测试 Db，湿热	25° C 到 55° C，湿度 95%
EN 60068-2-14，测试 Na，温度骤变	-40° C 到 +70° C，停留时间 3 小时，2 个周期
EN 60068-2-32，自由落体	0.3 m，5 次，产品包装
大气压	1080 到 660h Pa（相当于海拔 -1000 到 3500m）

环境条件 - 工作	
环境温度范围 （设备下部 25 mm 进风距离）	0° C 到 55° C，水平安装 0° C 到 45° C，垂直安装 湿度 95%，不结露
大气压	1080 到 795 hPa（相当于海拔 -1000 到 2000m）
污染物浓度	SO_2： < 0.5 ppm；H_2S： < 0.1 ppm；RH < 60%，不结露
EN 60068-2-14，测试 Nb，温度变化	5° C 至 55° C，3° C/分钟
EN 60068-2-27 机械冲击	15 G，11 ms 脉冲，3 个轴向上 6 次冲击
EN 60068-2-6 正弦振动	DIN 导轨安装：5-9 Hz 时 3.5 mm，9 - 150 Hz 时 1G 面板安装：5-9 Hz 时 7.0 mm，9 - 150 Hz 时 2G 每个轴 10 次摆动，每分 1 倍频程

高电位绝缘测试	
24 V/5 V 标称电路间	520 VDC（光隔离边界的型式测试）
115/230 V 电路对地	1500 VAC 常规测试/1950 VDC 型式测试
115/230 V 电路对 115/230 V 电路	1500 VAC 常规测试/1950 VDC 型式测试
115 V/230V 电路对 24 V/5 V 电路	1500 VAC 常规测试/3250 VDC 型式测试

保护类别

- 保护类别 II 符合 EN 61131-2（不需要保护导线）

防护等级

- IP20 机械保护，EN 60529
- 防止手指接触经标准探针测试出的高压。 需要针对灰尘、污物、水和直径小于 12.5mm 的异物施加外部保护。

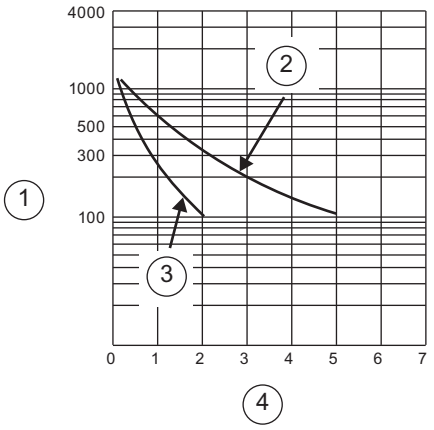
额定电压

额定电压	容错
24 VDC	20.4 VDC 到 28.8 VDC
120/230 VAC	85 VAC 到 264 VAC，47 到 63 Hz

注意
机械触点接通 S7-1200 CPU 的输出电源或任何数字量信号模块时，会发送信号“1”到数字量输出，时间约 50 微秒。 必须考虑这一点，尤其是使用响应短脉冲的设备时。

继电器电气使用寿命

继电器供应商提供的典型性能数据如下。 根据具体应用，实际性能可能不同。 使用适合于负载的外部保护电路可增强触点的使用寿命。



① 使用寿命（x 10^3 次动作）

② 250 VAC 阻性负载，
30 VDC 阻性负载

③ 250 VAC 感性负载 (p.f=0.4)
30 VDC 感性负载 (L/R=7ms)

④ 额定工作电流 (A)

A.2 CPU

A.2.1 CPU 1211C 规范

技术规范			
型号	**CPU 1211C AC/DC/继电器**	**CPU 1211C DC/DC/继电器**	**CPU 1211C DC/DC/DC**
订货号 (MLFB)	6ES7 211-1BD30-0XB0	6ES7 211-1HD30-0XB0	6ES7 211-1AD30-0XB0
常规			
尺寸 W x H x D (mm)	90 x 100 x 75		
重量	420 g	380 g	370 g
功耗	10 W	8 W	
可用电流（CM 总线）	最大 750 mA (5 VDC)		
可用电流 (24 VDC)	最大 300 mA（传感器电源）		
数字输入电流消耗 (24VDC)	所用的每点输入 4 mA		
CPU 特征			
用户存储器	25 KB 工作存储器/1 MB 装载存储器/2 KB 保持性存储器		
板载数字 I/O	6 点输入/4 点输出		
板载模拟 I/O	2 路输入		
过程映像大小	1024 字节输入 (I)/1024 字节输出 (Q)		
位存储器 (M)	4096 个字节		
信号模块扩展	无		
信号板扩展	最多 1 块信号板		
通信模块扩展	最多 3 个通信模块		
高速计数器	共 3 个 单相： 3 个，100 kHz 正交相位： 3 个，80 kHz		
脉冲输出	2		
脉冲捕捉输入	6		
延时中断/循环中断	共 4 个，精度为 1 ms		

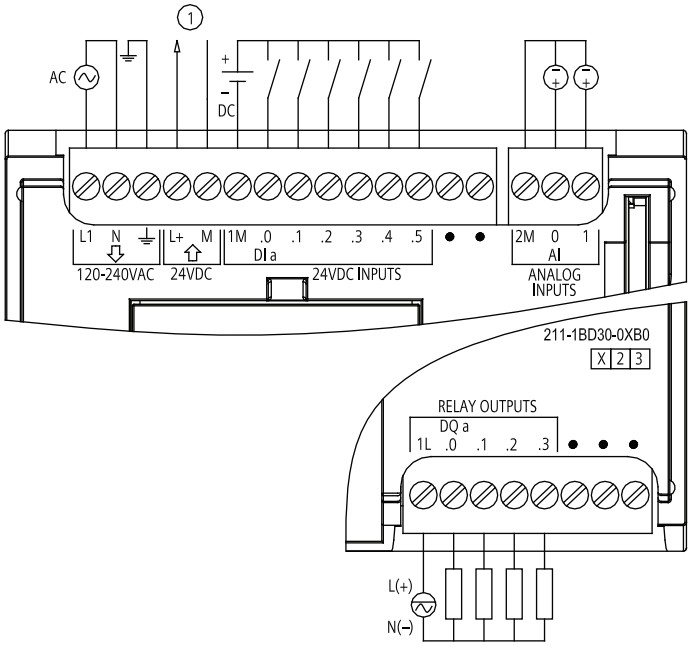
技术规范			
型号	**CPU 1211C AC/DC/继电器**	**CPU 1211C DC/DC/ 继电器**	**CPU 1211C DC/DC/DC**
沿中断	6 个上升沿和 6 个下降沿（使用可选信号板时，各为 10 个）		
存储卡	SIMATIC 存储卡（选件）		
实时时钟精度	+/- 60 秒/月		
实时时钟保持时间	通常为 10 天，40°C 时最少为 6 天（免维护超级电容）		
性能			
布尔运算执行速度	0.1 µs/指令		
移动字执行速度	12 µs/指令		
实数数学运算执行速度	18 µs/指令		
通信			
端口数	1		
类型	以太网		
连接数	• 3 个用于 HMI • 1 个用于编程设备 • 8 个用于用户程序中的以太网指令 • 3 个用于 CPU 对 CPU		
数据传输率	10/100 Mb/s		
隔离（外部信号与 PLC 逻辑侧）	变压器隔离，1500 VDC		
电缆类型	CAT5e 屏蔽电缆		
电源			
电压范围	85 到 264 VAC	20.4 到 28.8 VDC	
线路频率	47 到 63 Hz	--	
输入电流 最大负载时仅包括 CPU 最大负载时包括 CPU 和所有扩展附件	 120 VAC 时 60 mA 240 VAC 时 30 mA 120 VAC 时 180 mA 240 VAC 时 90 mA	 24 VDC 时 300 mA 24 VDC 时 900 mA	
突入电流（最大）	264 VAC 时 20 A	28.8 VDC 时 12 A	
隔离（输入电源与逻辑侧）	1500 VAC	未隔离	

技术规范			
型号	**CPU 1211C AC/DC/继电器**	**CPU 1211C DC/DC/ 继电器**	**CPU 1211C DC/DC/DC**
漏地电流，AC 线路对功能地	最大 0.5 mA	-	
保持时间（掉电）	120 VAC 时 20 ms 240 VAC 时 80 ms	24 VDC 时 10 ms	
内部保险丝，用户不可更换	3 A，250 V，慢速熔断		
传感器电源			
电压范围	20.4 到 28.8 VDC	L+ - 4 VDC（最小）	
额定输出电流（最大）	300 mA（短路保护）		
最大波纹噪声 (<10 MHz)	< 1 V 峰峰值	与输入线路相同	
隔离（CPU 逻辑侧与传感器电源）	未隔离		
数字输入			
输入路数	6		
类型	漏型/源型（IEC 1 类漏型）		
额定电压	4 mA 时 24 VDC，额定值		
允许的连续电压	最大 30 VDC		
浪涌电压	35 VDC，持续 0.5 s		
逻辑 1 信号（最小）	2.5 mA 时 15 VDC		
逻辑 0 信号（最大）	1 mA 时 5 VDC		
隔离（现场侧与逻辑侧）	500 VAC，持续 1 min		
隔离组	1		
滤波时间	0.2、0.4、0.8、1.6、3.2、6.4 和 12.8 ms（可选择，4 个为一组）		
HSC 时钟输入频率（最大）（逻辑 1 电平 = 15 到 26 VDC）	单相： 100 KHz 正交相位： 80 KHz		
同时接通的输入数	6		
电缆长度（米）	500（屏蔽）；300（非屏蔽）；50（屏蔽，HSC 输入）		
模拟输入			
输入路数	2		

技术规范			
型号	**CPU 1211C AC/DC/继电器**	**CPU 1211C DC/DC/继电器**	**CPU 1211C DC/DC/DC**
类型	电压（单侧）		
范围	0 到 10 V		
满量程范围（数据字）	0 到 27648（请参考模拟输入的电压表示法 (页 355)）		
过冲范围（数据字）	27,649 到 32,511（请参考模拟输入的电压表示法 (页 355)）		
溢出（数据字）	32,512 到 32767（请参考模拟输入的电压表示法 (页 355)）		
精度	10 位		
最大耐压	35 VDC		
平滑	无、弱、中或强（请参考模拟输入的响应时间 (页 355)以了解阶跃响应时间）		
噪声抑制	10、50 或 60 Hz（请参考模拟输入的响应时间 (页 355)以了解采样速率）		
阻抗	≥100 KΩ		
隔离（现场侧与逻辑侧）	无		
精度（25°C/0 到 55°C）	满量程的 3.0%/3.5%		
共模抑制	40 dB，DC 到 60 Hz		
工作信号范围	信号加共模电压必须小于 +12 V 且大于 -12 V		
电缆长度（米）	100 m，屏蔽双绞线		
数字输出			
输出点数	4		
类型	继电器，干触点		固态 - MOSFET
电压范围	5 到 30 VDC 或 5 到 250 VAC		20.4 到 28.8 VDC
最大电流时的逻辑 1 信号	--		最小 20 VDC
具有 10 KΩ 负载时的逻辑 0 信号	--		最大 0.1 VDC
电流（最大）	2.0 A		0.5 A
灯负载	30 W DC/200 W AC		5 W
通态电阻	新设备最大为 0.2 Ω		最大 0.6 Ω
每点的漏泄电流	--		最大 10 µA

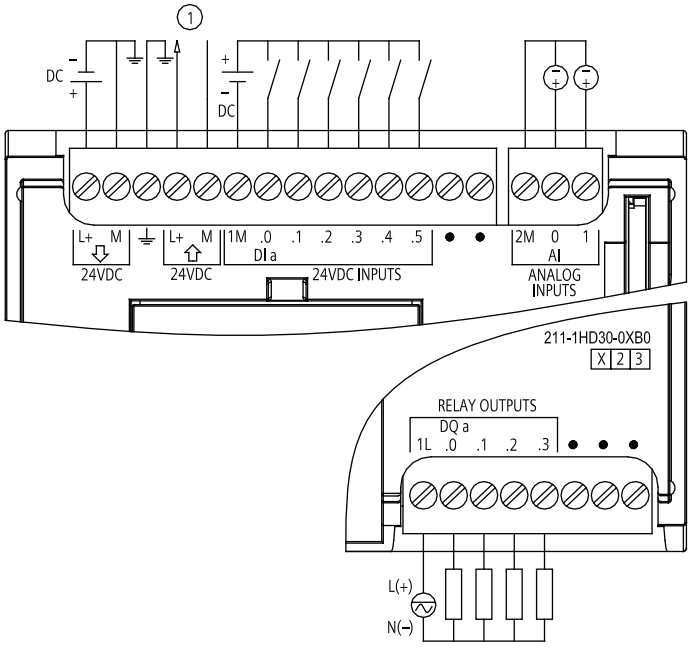
技术规范			
型号	**CPU 1211C AC/DC/继电器**	**CPU 1211C DC/DC/ 继电器**	**CPU 1211C DC/DC/DC**
浪涌电流	触点闭合时为 7 A		8 A，最长持续 100 ms
过载保护	无		
隔离（现场侧与逻辑侧）	1500 VAC，持续 1 min（线圈与触点） 无（线圈与逻辑侧）		500 VAC，持续 1 min
隔离电阻	新设备最小为 100 MΩ		--
断开触点间的绝缘	750 VAC，持续 1 min		--
隔离组	1		1
电感钳位电压	--		L+ - 48 VDC，1 W 损耗
开关延迟（Qa.0 到 Qa.3）	最长 10 ms		断开到接通最长为 1.0 µs 接通到断开最长为 3.0 µs
脉冲串输出频率（Qa.0 和 Qa.2）	不推荐		最大 100 KHz，最小 2 Hz
机械寿命（无负载）	10,000,000 个断开/闭合周期		--
额定负载下的触点寿命	100,000 个断开/闭合周期		--
RUN 到 STOP 时的行为	上一个值或替换值（默认值为 0）		
同时接通的输出数	4		
电缆长度（米）	500（屏蔽）；150（非屏蔽）		

接线图



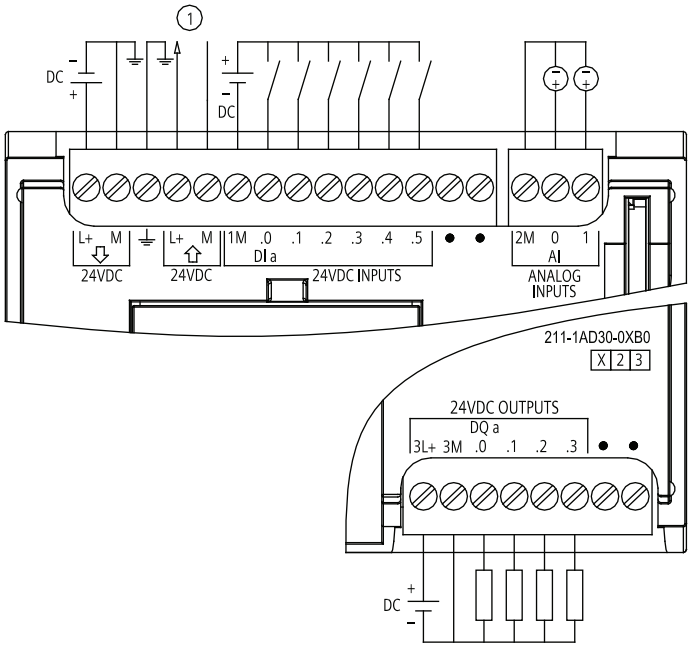
① 24 VDC 传感器电源输出

图 A-1 CPU 1211C AC/DC/继电器 (6ES7 211-1BD30-0XB0)



① 24 VDC 传感器电源输出

图 A-2 CPU 1211C DC/DC/继电器 (6ES7 211-1HD30-0XB0)



① 24 VDC 传感器电源输出

图 A-3 CPU 1211C DC/DC/DC (6ES7 211-1AD30-0XB0)

A.2.2 CPU 1212C 规范

技术规范			
型号	**CPU 1212C AC/DC/继电器**	**CPU 1212C DC/DC/继电器**	**CPU 1212C DC/DC/DC**
订货号 (MLFB)	6ES7 212-1BD30-0XB0	6ES7 212-1HD30-0XB0	6ES7 212-1AD30-0XB0
常规			
尺寸 W x H x D (mm)	90 x 100 x 75		
重量	425 g	385 g	370 g
功耗	11 W	9 W	
可用电流（SM 和 CM 总线）	最大 1000 mA (5 VDC)		
可用电流 (24 VDC)	最大 300 mA（传感器电源）		
数字输入电流消耗 (24 VDC)	所用的每点输入 4 mA		

技术规范			
型号	CPU 1212C AC/DC/继电器	CPU 1212C DC/DC/继电器	CPU 1212C DC/DC/DC
CPU 特征			
用户存储器	25 KB 工作存储器/1 MB 装载存储器/2 KB 保持性存储器		
板载数字 I/O	8 点输入/6 点输出		
板载模拟 I/O	2 路输入		
过程映像大小	1024 字节输入 (I)/1024 字节输出 (Q)		
位存储器 (M)	4096 个字节		
信号模块扩展	最多 2 个信号模块		
信号板扩展	最多 1 块信号板		
通信模块扩展	最多 3 个通信模块		
高速计数器	共 4 个 单相： 3 个 100 kHz 以及 1 个 30 kHz 的时钟频率 正交相位： 3 个 80 kHz 以及 1 个 20 kHz 的时钟频率		
脉冲输出	2		
脉冲捕捉输入	8		
延时中断/循环中断	共 4 个，精度为 1 ms		
沿中断	8 个上升沿和 8 个下降沿（使用可选信号板时，各为 12 个）		
存储卡	SIMATIC 存储卡（选件）		
实时时钟精度	+/- 60 秒/月		
实时时钟保持时间	通常为 10 天，40°C 时最少为 6 天（免维护超级电容）		
性能			
布尔运算执行速度	0.1 µs/指令		
移动字执行速度	12 µs/指令		
实数数学运算执行速度	18 µs/指令		
通信			
端口数	1		
类型	以太网		

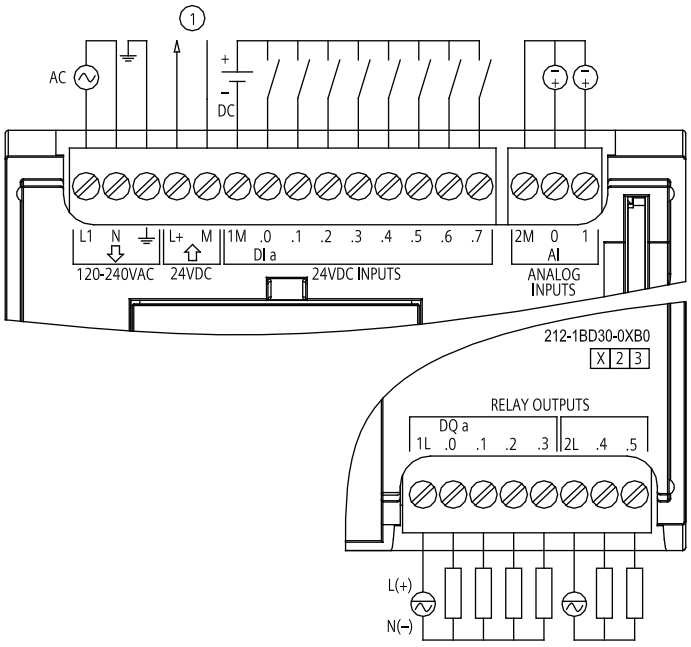
技术规范			
型号	**CPU 1212C AC/DC/继电器**	**CPU 1212C DC/DC/继电器**	**CPU 1212C DC/DC/DC**
连接数	• 3 个用于 HMI • 1 个用于编程设备 • 8 个用于用户程序中的以太网指令 • 3 个用于 CPU 对 CPU		
数据传输率	10/100 Mb/s		
隔离（外部信号与 PLC 逻辑侧）	变压器隔离，1500 VDC		
电缆类型	CAT5e 屏蔽电缆		
电源			
电压范围	85 到 264 VAC	20.4 到 28.8 VDC	
线路频率	47 到 63 Hz	--	
输入电流 最大负载时仅包括 CPU 最大负载时包括 CPU 和所有扩展附件	120 VAC 时 80 mA 240 VAC 时 40 mA 120 VAC 时 240 mA 240 VAC 时 120 mA	24 VDC 时 400 mA 24 VDC 时 1200 mA	
突入电流（最大）	264 VAC 时 20 A	28.8 VDC 时 12 A	
隔离（输入电源与逻辑侧）	1500 VAC	未隔离	
漏地电流，AC 线路对功能地	最大 0.5 mA	-	
保持时间（掉电）	120 VAC 时 20 ms 240 VAC 时 80 ms	24 VDC 时 10 ms	
内部保险丝，用户不可更换	3 A，250 V，慢速熔断		
传感器电源			
电压范围	20.4 到 28.8 VDC	L+ - 4 VDC（最小）	
额定输出电流（最大）	300 mA（短路保护）		
最大波纹噪声 (<10 MHz)	< 1 V 峰峰值	与输入线路相同	
隔离（CPU 逻辑侧与传感器电源）	未隔离		
数字输入			
输入点数	8		

技术规范			
型号	**CPU 1212C AC/DC/继电器**	**CPU 1212C DC/DC/继电器**	**CPU 1212C DC/DC/DC**
类型	漏型/源型（IEC 1 类漏型）		
额定电压	4 mA 时 24 VDC，额定值		
允许的连续电压	最大 30 VDC		
浪涌电压	35 VDC，持续 0.5 s		
逻辑 1 信号（最小）	2.5 mA 时 15 VDC		
逻辑 0 信号（最大）	1 mA 时 5 VDC		
隔离（现场侧与逻辑侧）	500 VAC，持续 1 min		
隔离组	1		
滤波时间	0.2、0.4、0.8、1.6、3.2、6.4 和 12.8 ms（可选择，4 个为一组）		
HSC 时钟输入频率（最大）（逻辑 1 电平 = 15 到 26 VDC）	单相： 100 KHz（Ia.0 到 Ia.5）和 30 KHz（Ia.6 到 Ia.7） 正交相位： 80 KHz（Ia.0 到 Ia.5）和 20 KHz（Ia.6 到 Ia.7）		
同时接通的输入数	8		
电缆长度（米）	500（屏蔽）；300（非屏蔽）；50（屏蔽，HSC 输入）		
模拟输入			
输入点数	2		
类型	电压（单侧）		
范围	0 到 10 V		
满量程范围（数据字）	0 到 27648（请参考模拟输入的电压表示法 (页 355)）		
过冲范围（数据字）	27,649 到 32,511（请参考模拟输入的电压表示法 (页 355)）		
溢出（数据字）	32,512 到 32767（请参考模拟输入的电压表示法 (页 355)）		
精度	10 位		
最大耐压	35 VDC		
平滑	无、弱、中或强（请参考模拟输入的响应时间 (页 355)以了解阶跃响应时间）		
噪声抑制	10、50 或 60 Hz（请参考模拟输入的响应时间 (页 355)以了解采样速率）		
阻抗	≥100 KΩ		

技术规范			
型号	**CPU 1212C AC/DC/继电器**	**CPU 1212C DC/DC/继电器**	**CPU 1212C DC/DC/DC**
隔离（现场侧与逻辑侧）	无		
精度（25°C/0 到 55°C）	满量程的 3.0%/3.5%		
共模抑制	40 dB，DC 到 60 Hz		
工作信号范围	信号加共模电压必须小于 +12 V 且大于 -12 V		
电缆长度（米）	100 米屏蔽双绞线		
数字输出			
输出点数	6		
类型	继电器，干触点		固态 - MOSFET
电压范围	5 到 30 VDC 或 5 到 250 VAC		20.4 到 28.8 VDC
最大电流时的逻辑 1 信号	--		最小 20 VDC
具有 10 KΩ 负载时的逻辑 0 信号	--		最大 0.1 VDC
电流（最大）	2.0 A		0.5 A
灯负载	30 W DC/200 W AC		5 W
通态电阻	新设备最大为 0.2 Ω		最大 0.6 Ω
每点的漏泄电流	--		最大 10 µA
浪涌电流	触点闭合时为 7 A		8 A，最长持续 100 ms
过载保护	无		
隔离（现场侧与逻辑侧）	1500 VAC，持续 1 min（线圈与触点） 无（线圈与逻辑侧）		500 VAC，持续 1 min
隔离电阻	新设备最小为 100 MΩ		--
断开触点间的绝缘	750 VAC，持续 1 min		--
隔离组	2		1
电感钳位电压	--		L+ - 48 VDC，1 W 损耗
开关延迟（Qa.0 到 Qa.3）	最长 10 ms		断开到接通最长为 1.0 µs 接通到断开最长为 3.0 µs
开关延迟（Qa.4 到 Qa.5）	最长 10 ms		断开到接通最长为 50 µs 接通到断开最长为 200 µs

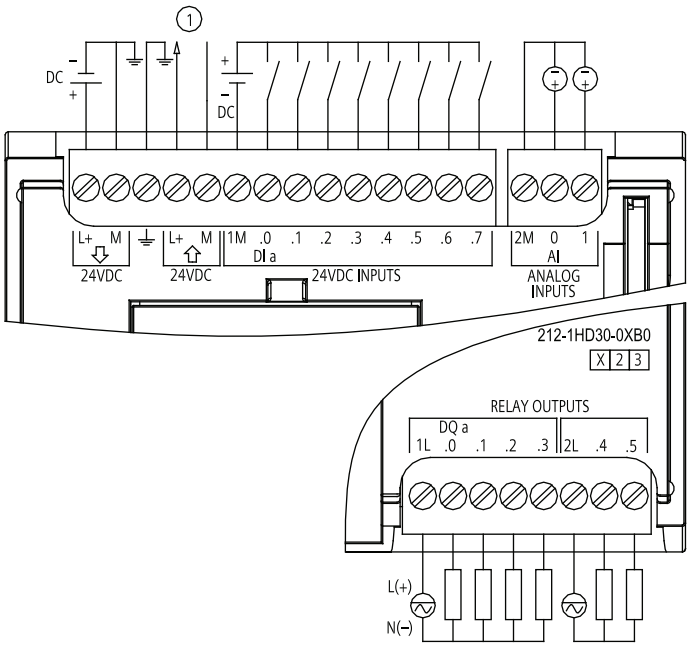
技术规范			
型号	**CPU 1212C AC/DC/继电器**	**CPU 1212C DC/DC/继电器**	**CPU 1212C DC/DC/DC**
脉冲串输出频率（Qa.0 和 Qa.2）	不推荐		最大 100 KHz，最小 2 Hz
机械寿命（无负载）	10,000,000 个断开/闭合周期		--
额定负载下的触点寿命	100,000 个断开/闭合周期		--
RUN 到 STOP 时的行为	上一个值或替换值（默认值为 0）		
同时接通的输出数	6		
电缆长度（米）	500（屏蔽）；150（非屏蔽）		

接线图



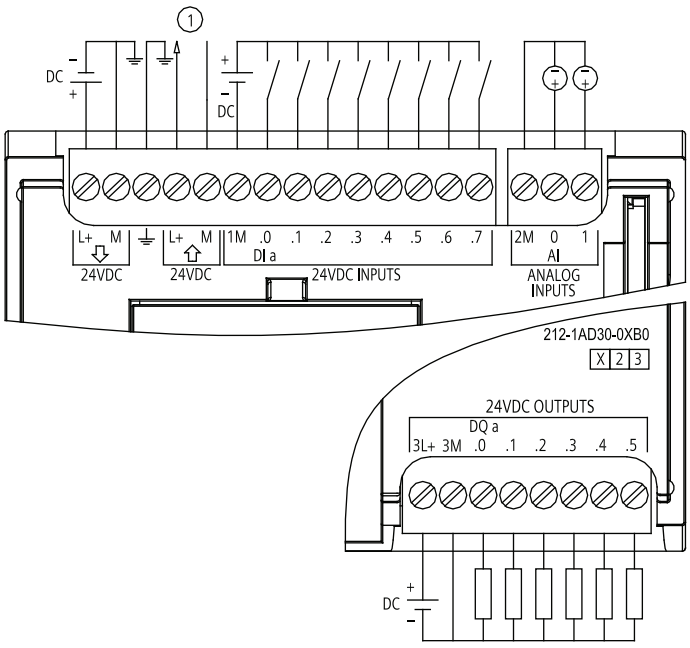
① 24 VDC 传感器电源输出

图 A-4 CPU 1212C AC/DC 继电器 (6ES7 212-1BD30-0XB0)



① 24 VDC 传感器电源输出

图 A-5 CPU 1212C DC/DC/继电器 (6ES7 212-1HD30-0XB0)



① 24 VDC 传感器电源输出

图 A-6 CPU 1212C DC/DC/DC (6ES7 212-1AD30-0XB0)

A.2.3 CPU 1214C 规范

技术规范			
型号	**CPU 1214C AC/DC/继电器**	**CPU 1214C DC/DC/继电器**	**CPU 1214C DC/DC/DC**
订货号 (MLFB)	6ES7 214-1BE30-0XB0	6ES7 214-1HE30-0XB0	6ES7 214-1AE30-0XB0
常规			
尺寸 W x H x D (mm)	110 x 100 x 75		
重量	475 g	435 g	415 g
功耗	14 W	12 W	
可用电流（SM 和 CM 总线）	最大 1600 mA (5 VDC)		
可用电流 (24 VDC)	最大 400 mA（传感器电源）		
数字输入电流消耗 (24VDC)	所用的每点输入 4 mA		
CPU 特征			
用户存储器	50 KB 工作存储器/2 MB 装载存储器/2 KB 保持性存储器		
板载数字 I/O	14 点输入/10 点输出		
板载模拟 I/O	2 路输入		
过程映像大小	1024 字节输入 (I)/1024 字节输出 (Q)		
位存储器 (M)	8192 个字节		
信号模块扩展	最多 8 个信号模块		
信号板扩展	最多 1 块信号板		
通信模块扩展	最多 3 个通信模块		
高速计数器	共 6 个 单相： 3 个 100 kHz 以及 3 个 30 kHz 的时钟频率 正交相位： 3 个 80 kHz 以及 3 个 20 kHz 的时钟频率		
脉冲输出	2		
脉冲捕捉输入	14		
延时中断/循环中断	共 4 个，精度为 1 ms		
沿中断	12 个上升沿和 12 个下降沿（使用可选信号板时，各为 14 个）		
存储卡	SIMATIC 存储卡（选件）		

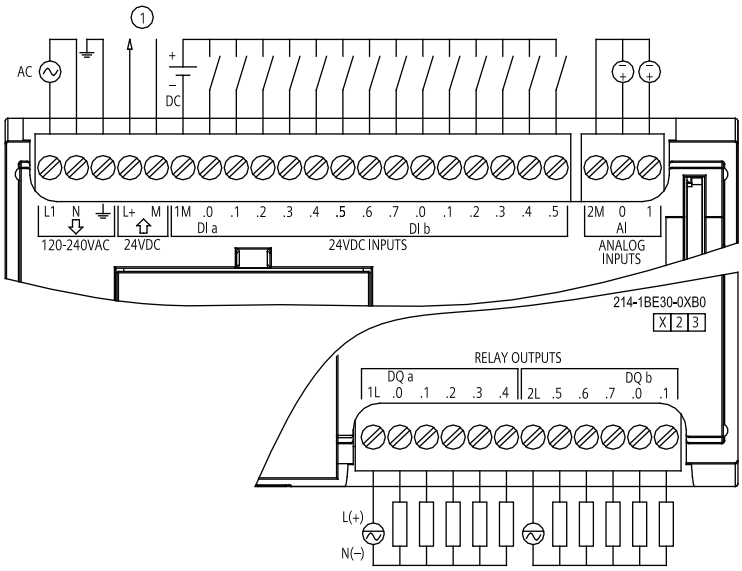
技术规范			
型号	**CPU 1214C AC/DC/继电器**	**CPU 1214C DC/DC/继电器**	**CPU 1214C DC/DC/DC**
实时时钟精度	+/- 60 秒/月		
实时时钟保持时间	通常为 10 天，40°C 时最少为 6 天（免维护超级电容）		
性能			
布尔运算执行速度	0.1 µs/指令		
移动字执行速度	12 µs/指令		
实数数学运算执行速度	18 µs/指令		
通信			
端口数	1		
类型	以太网		
连接数	• 3 个用于 HMI • 1 个用于编程设备 • 8 个用于用户程序中的以太网指令 • 3 个用于 CPU 对 CPU		
数据传输率	10/100 Mb/s		
隔离（外部信号与 PLC 逻辑侧）	变压器隔离，1500 VDC		
电缆类型	CAT5e 屏蔽电缆		
电源			
电压范围	85 到 264 VAC	20.4 到 28.8 VDC	
线路频率	47 到 63 Hz	--	
输入电流 最大负载时仅包括 CPU 最大负载时包括 CPU 和所有扩展附件	 120 VAC 时 100 mA 240 VAC 时 50 mA 120 VAC 时 300 mA 240 VAC 时 150 mA	 24 VDC 时 500 mA 24 VDC 时 1500 mA	
突入电流（最大）	264 VAC 时 20 A	28.8 VDC 时 12 A	
隔离（输入电源与逻辑侧）	1500 VAC	未隔离	
漏地电流，AC 线路对功能地	最大 0.5 mA	-	

技术规范			
型号	**CPU 1214C AC/DC/继电器**	**CPU 1214C DC/DC/继电器**	**CPU 1214C DC/DC/DC**
保持时间（掉电）	120 VAC 时 20 ms 240 VAC 时 80 ms	24 VDC 时 10 ms	
内部保险丝，用户不可更换	3 A，250 V，慢速熔断		
传感器电源			
电压范围	20.4 到 28.8 VDC	L+ - 4 VDC（最小）	
额定输出电流（最大）	400 mA（短路保护）		
最大波纹噪声 (<10 MHz)	< 1 V 峰峰值	与输入线路相同	
隔离（CPU 逻辑侧与传感器电源）	未隔离		
数字输入			
输入点数	14		
类型	漏型/源型（IEC 1 类漏型）		
额定电压	4 mA 时 24 VDC，额定值		
允许的连续电压	最大 30 VDC		
浪涌电压	35 VDC，持续 0.5 s		
逻辑 1 信号（最小）	2.5 mA 时 15 VDC		
逻辑 0 信号（最大）	1 mA 时 5 VDC		
隔离（现场侧与逻辑侧）	500 VAC，持续 1 min		
隔离组	1		
滤波时间	0.2、0.4、0.8、1.6、3.2、6.4 和 12.8 ms（可选择，4 个为一组）		
HSC 时钟输入频率（最大）（逻辑 1 电平 = 15 到 26 VDC）	单相： 100 KHz（Ia.0 到 Ia.5）和 30 KHz（Ia.6 到 Ib.5） 正交相位： 80 KHz（Ia.0 到 Ia.5）和 20 KHz（Ia.6 到 Ib.5）		
同时接通的输入数	14		
电缆长度（米）	500（屏蔽）；300（非屏蔽）；50（屏蔽，HSC 输入）		
模拟输入			
输入路数	2		
类型	电压（单侧）		

技术规范			
型号	**CPU 1214C AC/DC/继电器**	**CPU 1214C DC/DC/继电器**	**CPU 1214C DC/DC/DC**
范围	0 到 10 V		
满量程范围（数据字）	0 到 27648（请参考模拟输入的电压表示法 (页 355)）		
过冲范围（数据字）	27,649 到 32,511（请参考模拟输入的电压表示法 (页 355)）		
溢出（数据字）	32,512 到 32767（请参考模拟输入的电压表示法 (页 355)）		
精度	10 位		
最大耐压	35 VDC		
平滑	无、弱、中或强（请参考模拟输入的响应时间 (页 355)以了解阶跃响应时间）		
噪声抑制	10、50 或 60 Hz（请参考模拟输入的响应时间 (页 355)以了解采样速率）		
阻抗	≥100 KΩ		
隔离（现场侧与逻辑侧）	无		
精度（25°C/0 到 55°C）	满量程的 3.0%/3.5%		
共模抑制	40 dB，DC 到 60 Hz		
工作信号范围	信号加共模电压必须小于 +12 V 且大于 -12 V		
电缆长度（米）	100 米屏蔽双绞线		
数字输出			
输出点数	10		
类型	继电器，干触点		固态 - MOSFET
电压范围	5 到 30 VDC 或 5 到 250 VAC		20.4 到 28.8 VDC
最大电流时的逻辑 1 信号	--		最小 20 VDC
具有 10 KΩ 负载时的逻辑 0 信号	--		最大 0.1 VDC
电流（最大）	2.0 A		0.5 A
灯负载	30 W DC/200 W AC		5 W
通态电阻	新设备最大为 0.2 Ω		最大 0.6 Ω
每点的漏泄电流	--		最大 10 µA
浪涌电流	触点闭合时为 7 A		8 A，最长持续 100 ms

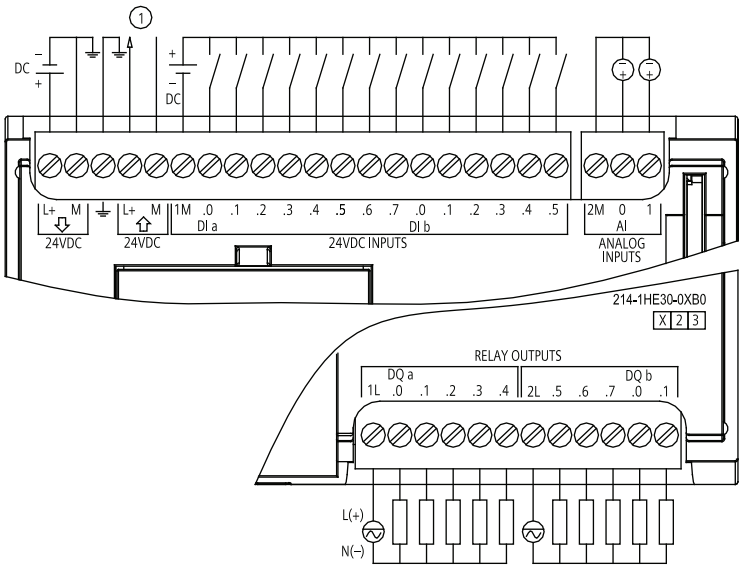
技术规范			
型号	**CPU 1214C AC/DC/继电器**	**CPU 1214C DC/DC/继电器**	**CPU 1214C DC/DC/DC**
过载保护	无		
隔离（现场侧与逻辑侧）	1500 VAC，持续 1 min（线圈与触点） 无（线圈与逻辑侧）		500 VAC，持续 1 min
隔离电阻	新设备最小为 100 MΩ		--
断开触点间的绝缘	750 VAC，持续 1 min		--
隔离组	2		1
电感钳位电压	--		L+ - 48 VDC，1 W 损耗
开关延迟（Qa.0 到 Qa.3）	最长 10 ms		断开到接通最长为 1.0 µs 接通到断开最长为 3.0 µs
开关延迟（Qa.4 到 Qb.1）	最长 10 ms		断开到接通最长为 50 µs 接通到断开最长为 200 µs
脉冲串输出频率（Qa.0 和 Qa.2）	不推荐		最大 100 KHz，最小 2 Hz
机械寿命（无负载）	10,000,000 个断开/闭合周期		--
额定负载下的触点寿命	100,000 个断开/闭合周期		--
RUN 到 STOP 时的行为	上一个值或替换值（默认值为 0）		
同时接通的输出数	10		
电缆长度（米）	500（屏蔽）；150（非屏蔽）		

接线图



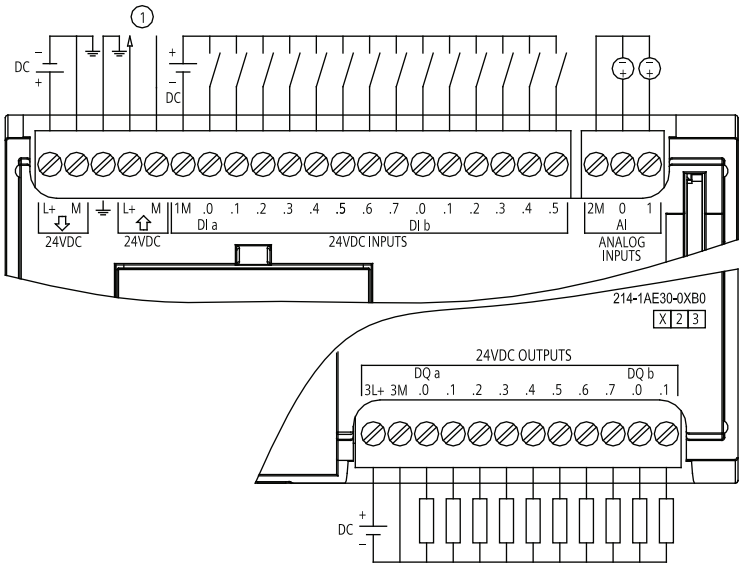
① 24 VDC 传感器电源输出

图 A-7 CPU 1214C AC/DC/继电器 (6ES7 214-1BE30-0XB0)



① 24 VDC 传感器电源输出

图 A-8 CPU 1214C DC/DC/继电器 (6ES7 214-1HE30-0XB0)



① 24 VDC 传感器电源输出

图 A-9 CPU 1214C DC/DC/DC (6ES7 214-1AE30-0XB0)

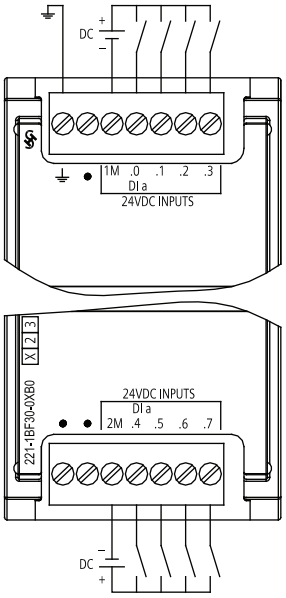
A.3 数字信号模块 (SM)

A.3.1 SM 1221 数字输入规范

技术规范		
型号	**SM 1221 DI 8x24VDC**	**SM 1221 DI 16x24VDC**
订货号 (MLFB)	6ES7 221-1BF30-0XB0	6ES7 221-1BH30-0XB0
常规		
尺寸 W x H x D (mm)	45 x 100 x 75	
重量	170 g	210 g
功耗	1.5 W	2.5 W
电流消耗（SM 总线）	105 mA	130 mA
电流消耗 (24 VDC)	所用的每点输入 4 mA	所用的每点输入 4 mA
数字输入		
输入点数	8	16
类型	漏型/源型（IEC 1 类漏型）	
额定电压	4 mA 时 24 VDC，额定值	
允许的连续电压	最大 30 VDC	
浪涌电压	35 VDC，持续 0.5 s	
逻辑 1 信号（最小）	2.5 mA 时 15 VDC	
逻辑 0 信号（最大）	1 mA 时 5 VDC	
隔离（现场侧与逻辑侧）	500 VAC，持续 1 min	
隔离组	2	4
滤波时间	0.2、0.4、0.8、1.6、3.2、6.4 和 12.8 ms（可选择，4 个为一组）	
同时接通的输入数	8	16
电缆长度（米）	500（屏蔽）；300（非屏蔽）	

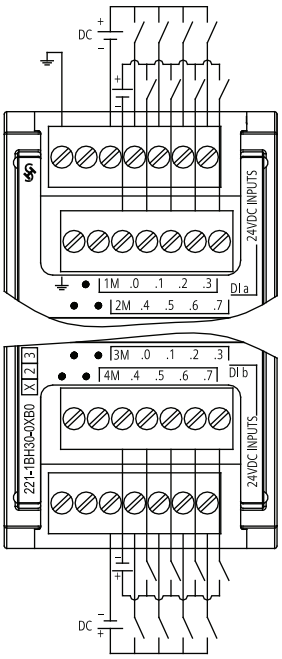
接线图

SM 1221 DI 8 x 24 VDC



6ES7 221-1BF30-0XB0

SM 1221 DI 16 x 24 VDC



6ES7 221-1BH30-0XB0

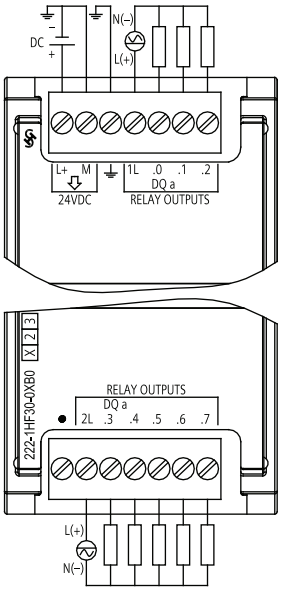
A.3.2 SM 1222 数字输出规范

技术规范				
型号	**SM 1222 DQ 8x 继电器**	**SM1222 DQ 16x 继电器**	**SM1222 DQ 8x24VDC**	**SM1222 DQ 16x24VDC**
订货号 (MLFB)	6ES7 222-1HF30-0XB0	6ES7 222-1HH30-0XB0	6ES7 222-1BF30-0XB0	6ES7 222-1BH30-0XB0
常规				
尺寸 W x H x D (mm)	45 x 100 x 75			
重量	190 g	260 g	180 g	220 g
功耗	4.5 W	8.5 W	1.5 W	2.5 W
电流消耗（SM 总线）	120 mA	135 mA	120 mA	140 mA
电流消耗 (24 VDC)	所用的每个继电器线圈 11 mA		--	
数字输出				
输出点数	8	16	8	16
类型	继电器，干触点		固态 - MOSFET	
电压范围	5 到 30 VDC 或 5 到 250 VAC		20.4 到 28.8 VDC	
最大电流时的逻辑 1 信号	--		最小 20 VDC	
具有 10 KΩ 负载时的逻辑 0 信号	--		最大 0.1 VDC	
电流（最大）	2.0 A		0.5 A	
灯负载	30 W DC/200 W AC		5W	
通态触点电阻	新设备最大为 0.2 Ω		最大 0.6 Ω	
每点的漏泄电流	--		最大 10 µA	
浪涌电流	触点闭合时为 7 A		8 A，最长持续 100 ms	
过载保护	无			
隔离（现场侧与逻辑侧）	1500 VAC，持续 1 min（线圈与触点） 无（线圈与逻辑侧）		500 VAC，持续 1 min	
隔离电阻	新设备最小为 100 MΩ		--	
断开触点间的绝缘	750 VAC，持续 1 min		--	
隔离组	2	4	1	1

技术规范				
型号	SM 1222 DQ 8x 继电器	SM1222 DQ 16x 继电器	SM1222 DQ 8x24VDC	SM1222 DQ 16x24VDC
每个公共端的电流（最大）	10 A		4 A	8 A
电感钳位电压	--		L+ - 48 V，1 W 损耗	
开关延迟	最长 10 ms		断开到接通最长为 50 µs 接通到断开最长为 200 µs	
机械寿命（无负载）	10,000,000 个断开/闭合周期		--	
额定负载下的触点寿命	100,000 个断开/闭合周期		--	
RUN 到 STOP 时的行为	上一个值或替换值（默认值为 0）			
同时接通的输出数	8	16	8	16
电缆长度（米）	500（屏蔽）；150（非屏蔽）			

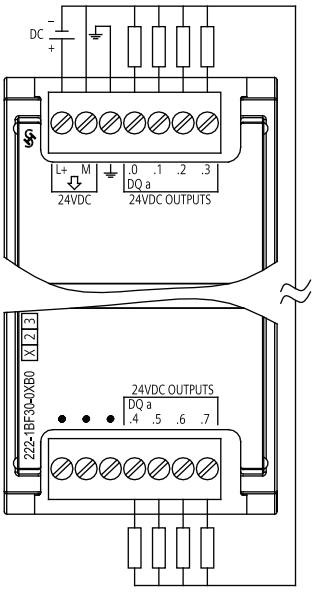
接线图

SM 1222 DQ 8 x 继电器



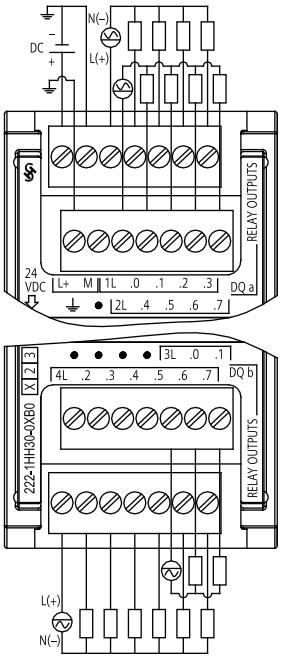
6ES7 222-1HF30-0XB0

SM 1222 DQ 8 x 24 VDC



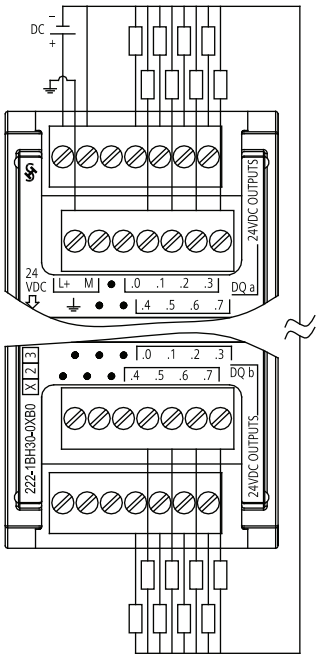
6ES7 222-1BF30-0XB0

SM 1222 DQ 16 x 继电器



6ES7 222-1HH30-0XB0

SM 1222 DQ 16 x 24 VDC



6ES7 222-1BH30-0XB0

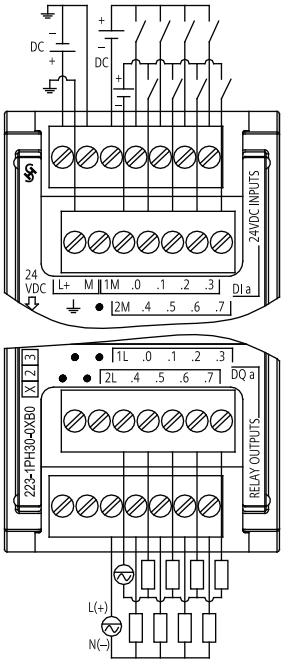
A.3.3 SM 1223 数字输入/输出规范

技术规范				
型号	**SM 1223 DI 8x24 VDC，DQ 8x 继电器**	**SM 1223 DI 16x24 VDC，DQ 16x 继电器**	**SM 1223 DI 8x24 VDC，DQ 8x24 VDC**	**SM 1223 DI 16x24 VDC，DQ16x24 VDC**
订货号 (MLFB)	6ES7 223-1PH30-0XB0	6ES7 223-1PL30-0XB0	6ES7 223-1BH30-0XB0	6ES7 223-1BL30-0XB0
尺寸 W x H x D (mm)	45 x 100 x 75	70 x 100 x 75	45 x 100 x 75	70 x 100 x 75
重量	230 g	350 g	210 g	310 g
功耗	5.5 W	10 W	2.5 W	4.5 W
电流消耗（SM 总线）	145 mA	180 mA	145 mA	185 mA
电流消耗 (24 VDC)	所用的每点输入 4 mA 所用的每个继电器线圈 11 mA		所用的每点输入 4 mA	
数字输入				
输入点数	8	16	8	16
类型	漏型/源型（IEC 1 类漏型）			
额定电压	4 mA 时 24 VDC，额定值			
允许的连续电压	最大 30 VDC			
浪涌电压	35 VDC，持续 0.5 s			
逻辑 1 信号（最小）	2.5 mA 时 15 VDC			
逻辑 0 信号（最大）	1 mA 时 5 VDC			
隔离（现场侧与逻辑侧）	500 VAC，持续 1 min			
隔离组	2	2	2	2
滤波时间	0.2、0.4、0.8、1.6、3.2、6.4 和 12.8 ms（可选择，4 个为一组）			
同时接通的输入数	8	16	8	16
电缆长度（米）	500（屏蔽）；300（非屏蔽）			
数字输出				
输出点数	8	16	8	16
类型	继电器，干触点		固态 - MOSFET	

技术规范				
型号	SM 1223 DI 8x24 VDC，DQ 8x 继电器	SM 1223 DI 16x24 VDC，DQ 16x 继电器	SM 1223 DI 8x24 VDC，DQ 8x24 VDC	SM 1223 DI 16x24 VDC，DQ16x24 VDC
电压范围	5 到 30 VDC 或 5 到 250 VAC		20.4 到 28.8 VDC	
最大电流时的逻辑 1 信号	--		最小 20 VDC	
具有 10 KΩ 负载时的逻辑 0 信号	--		最大 0.1 VDC	
电流（最大）	2.0 A		0.5 A	
灯负载	30 W DC/200 W AC		5 W	
通态触点电阻	新设备最大为 0.2 Ω		最大 0.6 Ω	
每点的漏泄电流	--		最大 10 µA	
浪涌电流	触点闭合时为 7 A		8 A，最长持续 100 ms	
过载保护	无			
隔离（现场侧与逻辑侧）	1500 VAC，持续 1 min（线圈与触点） 无（线圈与逻辑侧）		500 VAC，持续 1 min	
隔离电阻	新设备最小为 100 MΩ		--	
断开触点间的绝缘	750 VAC，持续 1 min		--	
隔离组	2	4	1	1
每个公共端的电流	10A	8 A	4 A	8 A
电感钳位电压	--		L+ - 48 V，1 W 损耗	
开关延迟	最长 10 ms		断开到接通最长为 50 µs 接通到断开最长为 200 µs	
机械寿命（无负载）	10,000,000 个断开/闭合周期		--	
额定负载下的触点寿命	100,000 个断开/闭合周期		--	
RUN 到 STOP 时的行为	上一个值或替换值（默认值为 0）			
同时接通的输出数	8	16	8	16
电缆长度（米）	500（屏蔽）；150（非屏蔽）			

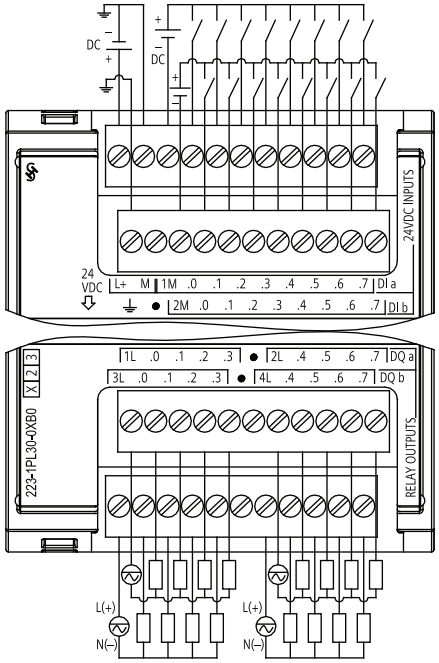
接线图

SM 1223 DI 8 x 24 VDC，DQ 8 x 继电器



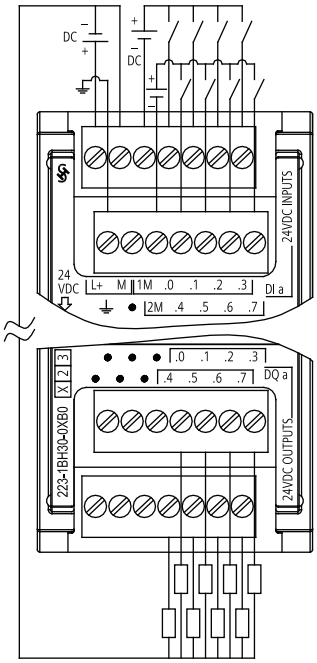
6ES7 223-1PH30-0XB0

SM1223 DI 16 x 24 VDC，DQ 16 x 继电器



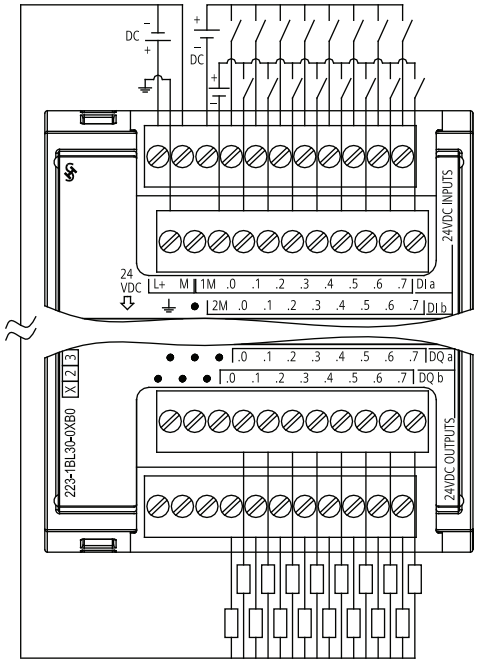
6ES7 223-1PL30-0XB0

SM 1223 DI 8 x 24 VDC，DQ 8 x 24 VDC



6ES7 223-1BH30-0XB0

SM 1223 DI 16 x 24 VDC，DQ 16 x 24 VDC



6ES7 223-1BL30-0XB0

A.4 模拟信号模块 (SM)

A.4.1 SM 1231、SM 1232、SM 1234 模拟量规范

技术规范			
型号	**SM 1231 AI 4x13 位**	**SM 1231 AI 8x13 位**	**SM 1234 AI 4x13 位 AQ 2x14 位**
订货号 (MLFB)	6ES7 231-4HD30-0XB0	6ES7 231-4HF30-0XB0	6ES7 234-4HE30-0XB0
常规			
尺寸 W x H x D (mm)	45 x 100 x 75	45 x 100 x 75	45 x 100 x 75
重量	180 g	180 g	220 g
功耗	1.5 W	1.5 W	2.0 W
电流消耗（SM 总线）	80 mA	90 mA	80 mA
电流消耗 (24 VDC)	45 mA	45 mA	60 mA（无负载）
模拟输入			
输入路数	4	8	4
类型	电压或电流（差动）： 可 2 个选为一组		
范围	±10 V、±5 V、±2.5 V 或 0 到 20 mA		
满量程范围（数据字）	-27,648 到 27,648		
过冲/下冲范围（数据字）	电压： 32,511 到 27,649/-27,649 到 -32,512 电流： 32,511 到 27,649/0 到 -4864 （请参考模拟输入的电压表示法，模拟输入的电流表示法 (页 355)）		
上溢/下溢（数据字）	电压： 32,767 到 32,512/-32,513 到 -32,768 电流： 32,767 到 32,512/-4865 到 -32,768 （请参考模拟输入的电压表示法，模拟输入的电流表示法 (页 355)）		
精度	12 位 + 符号位		
最大耐压/耐流	±35 V/±40 mA		
平滑	无、弱、中或强（请参考模拟输入的响应时间 (页 355)以了解阶跃响应时间）		

技术规范			
型号	**SM 1231 AI 4x13 位**	**SM 1231 AI 8x13 位**	**SM 1234 AI 4x13 位 AQ 2x14 位**
噪声抑制	400、60、50 或 10 Hz（请参考模拟输入的响应时间 (页 355)以了解采样速率）		
阻抗	≥ 9 MΩ（电压）/250 Ω（电流）		
隔离（现场侧与逻辑侧）	无		
精度（25°C/0 到 55°C）	满量程的 ±0.1%/±0.2%		
模数转换时间	625 µs（400 Hz 抑制）		
共模抑制	40 dB，DC 到 60 Hz		
工作信号范围	信号加共模电压必须小于 +12 V 且大于 -12 V		
电缆长度（米）	100 米，屏蔽双绞线		
诊断			
上溢/下溢	是 1)	有 1)	有 1)
对地短路（仅限电压模式）	不适用	不适用	输出端有
断路（仅限电流模式）	不适用	不适用	输出端有
24 VDC 低压	有	有	有

1 如果对输入端施加大于 +30 VDC 或小于 -15 VDC 的电压，则结果值将是未知的，因此相应的上溢或下溢可能不会激活。

技术规范			
型号	**SM 1232 AQ 2x14 位**	**SM 1232 AQ 4x14 位**	**SM 1234 AI 4x13 位 AQ 2x14 位**
订货号 (MLFB)	6ES7 232-4HB30-0XB0	6ES7 232-4HD30-0XB0	6ES7 234-4HE30-0XB0
常规			
尺寸 W x H x D (mm)	45 x 100 x 75	45 x 100 x 75	45 x 100 x 75
重量	180 g	180 g	220 g
功耗	1.5 W	1.5 W	2.0 W
电流消耗（SM 总线）	80 mA	80 mA	80 mA
电流消耗 (24 VDC)	45 mA（无负载）	45 mA（无负载）	60 mA（无负载）

技术规范			
型号	**SM 1232 AQ 2x14 位**	**SM 1232 AQ 4x14 位**	**SM 1234 AI 4x13 位 AQ 2x14 位**
模拟输出			
输出路数	2	4	2
类型	电压或电流		
范围	±10 V 或 0 到 20 mA		
精度	电压： 14 位；电流： 13 位		
满量程范围（数据字）	电压： -27,648 到 27,648；电流： 0 到 27,648 （请参考模拟量输出的电压表示法，模拟量输出的电流表示法 (页 355)）		
精度（25°C/0 到 55°C）	满量程的 ±0.3%/±0.6%		
稳定时间（新值的 95%）	电压： 300 µS (R)、750 µS (1 uF)；电流： 600 µS (1 mH)、2 ms (10 mH)		
负载阻抗	电压： ≥ 1000 Ω；电流： ≤ 600 Ω		
RUN 到 STOP 时的行为	上一个值或替换值（默认值为 0）		
隔离（现场侧与逻辑侧）	无		
电缆长度（米）	100 米，屏蔽双绞线		
诊断			
上溢/下溢	有	有	有 [1)]
对地短路（仅限电压模式）	有	有	输出端有
断路（仅限电流模式）	有	有	输出端有
24 VDC 低压	有	有	有

1 如果对输入端施加大于 +30 VDC 或小于 -15 VDC 的电压，则结果值将是未知的，因此相应的上溢或下溢可能不会激活。

模拟输入的响应时间

SM 模拟模块的阶跃响应 (ms)				
0V 到 10V，在 95% 时测得				
平滑选项	抑制频率			
	400 Hz	60 Hz	50 Hz	10 Hz
无	4	18	22	100
弱	9	52	63	320
中等	32	203	241	1200
强	61	400	483	2410
采样速率 • 4 个通道 • 8 个通道	 • 0.625 • 1.25	 • 4.17 • 4.17	 • 5 • 5	 • 25 • 25

CPU 模拟输入的阶跃响应 (ms)			
0V 到 10V，在 95% 时测得			
平滑选项	抑制频率		
	60 Hz	50 Hz	10 Hz
无	63	65	130
弱	84	93	340
中等	221	258	1210
强	424	499	2410
采样速率	**4.17**	**5**	**25**

模拟输入的电压表示法

体系	电压测量范围						
十进制	十六进制	±10 V	±5 V	±2.5 V		0 到 10 V	
32767	7FFF	11.851 V	5.926 V	2.963 V	上溢	11.851V	上溢
32512	7F00						
32511	7EFF	11.759 V	5.879 V	2.940 V	过冲范围	11.759 V	过冲范围
27649	6C01						
27648	6C00	10 V	5 V	2.5 V	额定范围	10 V	额定范围
20736	5100	7.5 V	3.75 V	1.875 V		7.5 V	
1	1	361.7 µV	180.8 µV	90.4 µV		361.7 µV	
0	0	0 V	0 V	0 V		0 V	
-1	FFFF					不支持负值	
-20736	AF00	-7.5 V	-3.75 V	-1.875 V			
-27648	9400	-10 V	-5 V	-2.5 V			
-27649	93FF				下冲范围		
-32512	8100	-11.759 V	-5.879 V	-2.940 V			
-32513	80FF				下溢		
-32768	8000	-11.851 V	-5.926 V	-2.963 V			

模拟输入的电流表示法

体系	电流测量范围		
十进制	十六进制	0 mA 到 20 mA	
32767	7FFF	23.70 mA	上溢
32512	7F00		
32511	7EFF	23.52 mA	过冲范围
27649	6C01		
27648	6C00	20 mA	额定范围
20736	5100	15 mA	
1	1	723.4 nA	
0	0	0 mA	
-1	FFFF		下冲范围
-4864	ED00	-3.52 mA	
-4865	ECFF		下溢
-32768	8000		

模拟输出的电压表示法

体系		电压输出范围	
十进制	十六进制	±10 V	
32767	7FFF	请参见注 1	上溢
32512	7F00	请参见注 1	
32511	7EFF	11.76 V	过冲范围
27649	6C01		
27648	6C00	10 V	额定范围
20736	5100	7.5 V	
1	1	361.7 µV	
0	0	0 V	
-1	FFFF	-361.7 µV	
-20736	AF00	-7.5 V	
-27648	9400	-10 V	
-27649	93FF		下冲范围
-32512	8100	-11.76 V	
-32513	80FF	请参见注 1	下溢
-32768	8000	请参见注 1	

1 . 在下溢或上溢情况下，模拟量输出将根据为模拟量信号模块设置的设备配置属性动作。在“对 CPU STOP 的响应”(Reaction to CPU STOP) 参数中，任选“使用替换值”(Use substitute value) 或“保持上一个值”(Keep last value)。

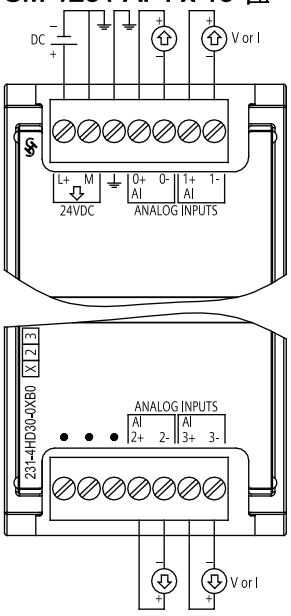
模拟输出的电流表示法

体系		电流输出范围	
十进制	十六进制	±20 mA	
32767	7FFF	请参见注 1	上溢
32512	7F00	请参见注 1	
32511	7EFF	23.52 mA	过冲范围
27649	6C01		
27648	6C00	20 mA	额定范围
20736	5100	15 mA	
1	1	723.4 nA	
0	0	0 mA	
-1	FFFF		下冲范围
-32512	8100		
-32513	80FF	请参见注 1	下溢
-32768	8000	请参见注 1	

1. 在下溢或上溢情况下，模拟量输出将根据为模拟量信号模块设置的设备配置属性动作。 在“对 CPU STOP 的响应”(Reaction to CPU STOP) 参数中，任选“使用替换值”(Use substitute value) 或“保持上一个值”(Keep last value)。

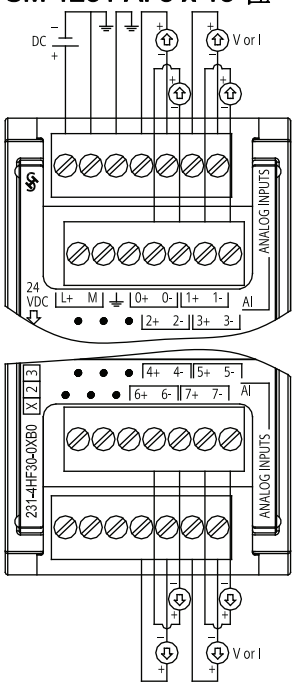
接线图

SM 1231 AI 4 x 13 位



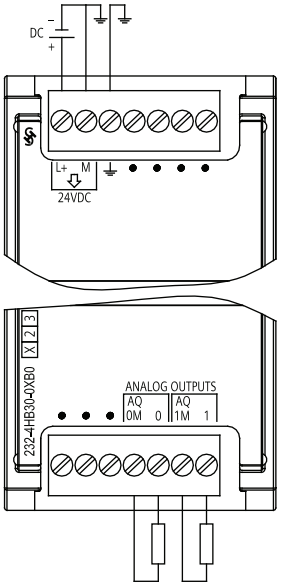
6ES7 231-4HD30-0XB0

SM 1231 AI 8 x 13 位



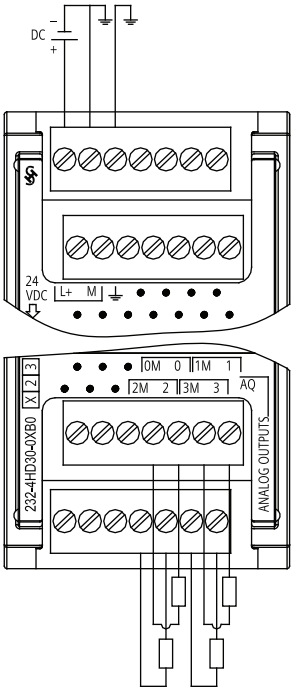
6ES7 231-4HF30-0XB0

SM 1232 AQ 2 x 14 位



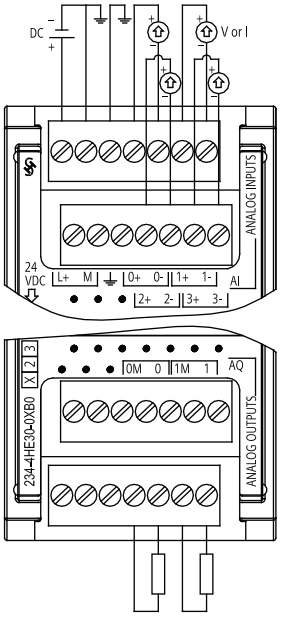
6ES7 232-4HB30-0XB0

SM 1232 AQ 4 x 14 位



6ES7 232-4HD30-0XB0

SM 1234 AI 4 x 13 位/ AQ 2 x 14 位



6ES7 234-4HE30-0XB0

A.5 信号板 (SB)

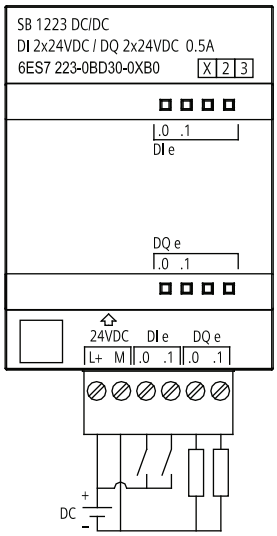
A.5.1 SB 1223 2 X 24 VDC 输入/2 X 24 VDC 输出规范

数字信号板规范

技术数据	
型号	**SB 1223 DI 2x24VDC，DQ 2x24VDC**
订货号 (MLFB)	6ES7 223-0BD30-0XB0
常规	
尺寸 W x H x D (mm)	38 x 62 x 21
重量	40 g
功耗	1.0 W
电流消耗（SM 总线）	50 mA
电流消耗 (24 VDC)	所用的每点输入 4 mA
数字输入	
输入点数	2
类型	IEC 1 类漏型
额定电压	4 mA 时 24 VDC，额定值
允许的连续电压	最大 30 VDC
浪涌电压	35 VDC，持续 0.5 s
逻辑 1 信号（最小）	2.5 mA 时 15 VDC
逻辑 0 信号（最大）	1 mA 时 5 VDC
HSC 时钟输入频率（最大）	20 kHz（15 到 30 VDC） 30 kHz（15 到 26 VDC）
隔离（现场侧与逻辑侧）	500 VAC，持续 1 min
隔离组	1
滤波时间	0.2、0.4、0.8、1.6、3.2、6.4 和 12.8 ms 可选择，2 个为一组
同时接通的输入数	2

技术数据	
型号	**SB 1223 DI 2x24VDC，DQ 2x24VDC**
电缆长度（米）	500（屏蔽）；300（非屏蔽）
数字输出	
输出点数	2
输出类型	固态 - MOSFET
电压范围	20.4 到 28.8 VDC
最大电流时的逻辑 1 信号	最小 20 VDC
具有 10 KΩ 负载时的逻辑 0 信号	最大 0.1 VDC
电流（最大）	0.5 A
灯负载	5 W
通态触点电阻	最大 0.6 Ω
每点的漏泄电流	最大 10 μA
脉冲串输出频率	最大 20 KHz，最小 2 Hz
浪涌电流	5 A，最长持续 100 ms
过载保护	无
隔离（现场侧与逻辑侧）	500 VAC，持续 1 min
隔离组	1
每个公共端的电流	1 A
电感钳位电压	L+ - 48 V，1 W 损耗
开关延迟	断开到接通最长为 2 μs 接通到断开最长为 10 μs
RUN 到 STOP 时的行为	上一个值或替换值（默认值为 0）
同时接通的输出数	2
电缆长度（米）	500（屏蔽）；150（非屏蔽）

SB 1223 2 x 24 VDC 输入/2 x 24 VDC 输出接线图



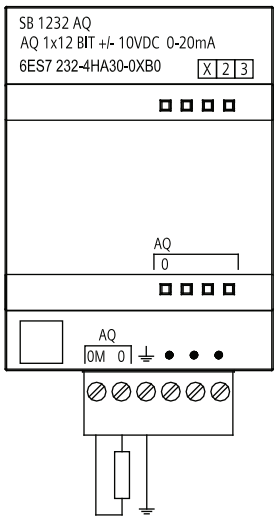
A.5.2 SB 1232 1 路模拟量输出规范

模拟信号板规范

技术数据	
型号	**SB 1223 AQ 1x12 位**
订货号 (MLFB)	6ES7 232-4HA30-0XB0
常规	
尺寸 W x H x D (mm)	38 x 62 x 21 mm
重量	40 g
功耗	1.5 W
电流消耗（SM 总线）	15 mA
电流消耗 (24 VDC)	40 mA（无负载）
模拟输出	
输出路数	1
类型	电压或电流
范围	±10 V 或 0 到 20 mA
精度	电压： 12 位 电流：11 位
满量程范围（数据字）	电压： -27,648 到 27,648 电流：0 到 27,648
精度（25°C/0 到 55°C）	满量程的 ±0.5%/±1%
稳定时间（新值的 95%）	电压： 300 µS (R)、750 µS (1 uF) 电流： 600 µS (1 mH)、2 ms (10 mH)
负载阻抗	电压： ≥ 1000 Ω 电流： ≤ 600 Ω
RUN 到 STOP 时的行为	上一个值或替换值（默认值为 0）
隔离（现场侧与逻辑侧）	无
电缆长度（米）	100 米，屏蔽双绞线
诊断	
上溢/下溢	有

技术数据	
型号	**SB 1223 AQ 1x12 位**
对地短路（仅限电压模式）	有
断路（仅限电流模式）	有

SB 1232 1 x 模拟量输出接线图



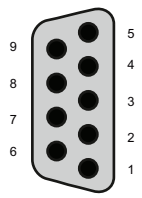
A.6 通信模块 (CM)

A.6.1 CM 1241 RS485 规范

表格 A- 1 通信模块 CM 1241 RS485

技术数据	
订货号 (MLFB)	6ES7 241-1CH30-0XB0
尺寸和重量	
尺寸	30 x 100 x 75 mm
重量	150 g
发送器和接收器	
共模电压范围	-7 V 到 +12 V，1 秒，3 VRMS 连续

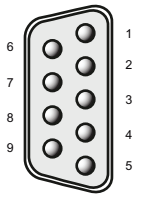
技术数据	
发送器差动输出电压	R_L = 100 Ω 时最小 2 V， R_L = 54 Ω 时最小 1.5 V
终端和偏置	B 上 10K Ω 对 +5 V，PROFIBUS 针 3 A 上 10K Ω 对 GND，PROFIBUS 针 8
接收器输入阻抗	最小 5.4K Ω，包括终端
接收器阈值/灵敏度	最低 +/- 0.2 V，典型滞后 60 mV
隔离 RS485 信号与外壳接地 RS485 信号与 CPU 逻辑公共端	500 VAC，1 分钟
电缆长度，屏蔽电缆	最长 1000 m
电源规范	
功率损失（损耗）	1.1 W
+5 VDC 电流	220 mA

引脚	说明	连接器（母）	引脚	说明
1 GND	逻辑地或通信地		6 PWR	+5V 与 100 Ω 串联电阻： 输出
2	未连接		7	未连接
3 TxD+	信号 B (RxD/TxD+)： 输入/输出		8 TXD-	信号 A (RxD/TxD-)： 输入/输出
4 RTS	请求发送（TTL 电平）：输出		9	未连接
5 GND	逻辑地或通信地		SHELL	外壳接地

A.6.2 CM 1241 RS232 规范

通信模块 CM 1241 RS232

技术数据	
订货号 (MLFB)	6ES7 241-1AH30-0XB0
尺寸和重量	
尺寸	30 x 100 x 75 mm
重量	150 g
发送器和接收器	
发送器输出电压	R_L = 3K Ω 时最小 +/- 5 V
传送输出电压	最大 +/- 15 VDC
接收器输入阻抗	最小 3 K Ω
接收器阈值/灵敏度	最低 0.8 V，最高 2.4 V 典型滞后 0.5 V
接收器输入电压	最大 +/- 30VDC
隔离 RS 232 信号与外壳接地 RS 232 信号与 CPU 逻辑公共端	500 VAC，1 分钟
电缆长度，屏蔽电缆	最长 10 m
电源规范	
功率损失（损耗）	1.1 W
+5 VDC 电流	220 mA

引脚	说明	连接器（公）	引脚	说明
1 DCD	数据载波检测： 输入		6 DSR	数据设备就绪： 输入
2 RxD	从 DCE 接收数据： 输入		7 RTS	请求发送： 输出
3 TxD	传送数据到 DCE： 输出		8 CTS	允许发送： 输入
4 DTR	数据终端就绪： 输出		9 RI	振铃指示器（未用）
5 GND	逻辑地		SHELL	外壳接地

A.7 SIMATIC 存储卡

存储卡规范

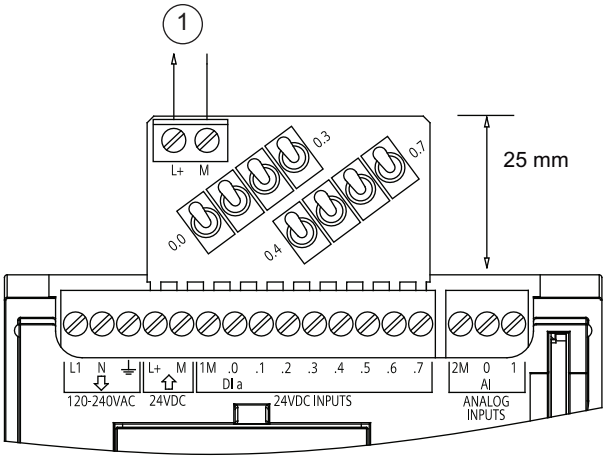
订货号	容量
6ES7 954-8LF00-0AA0	24 MB
6ES7 954-8LB00-0AA0	2 MB

A.8 输入仿真器

型号	8 位置仿真器	14 位置仿真器
订货号 (MLFB)	6ES7 274-1XF30-0XA0	6ES7 274-1XH30-0XA0
尺寸 W x H x D (mm)	43 x 35 x 23	67 x 35 x 23
重量	20 g	30 g
点数	8	14
配套使用的 CPU	CPU 1211C、CPU 1212C	CPU 1214C

⚠警告
这些输入仿真器未获准在 Class I DIV 2 或 Class I Zone 2 危险场所使用。 如果在 Class I DIV 2 或 Class I Zone 2 场所使用，开关存在潜在的打火危险/爆炸危险。

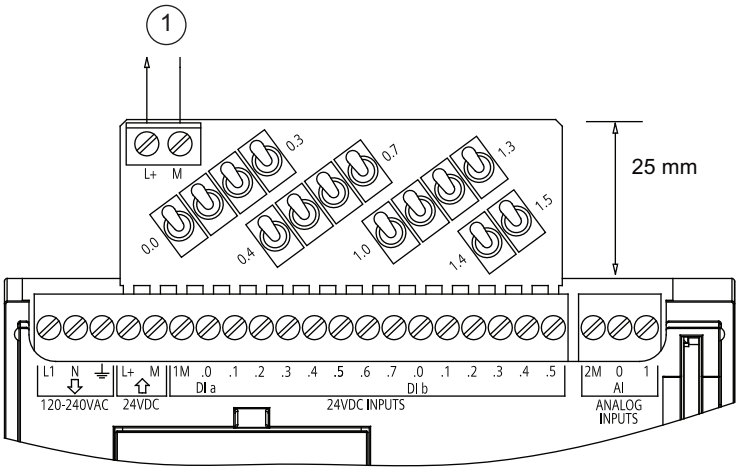
8 位置仿真器



① 24 VDC 传感器电源输出

6ES7 274-1XF30-0XA0

14 位置仿真器



① 24 VDC 传感器电源输出

6ES7 274-1XH30-0XA0

A.9 I/O 扩展电缆

技术数据	
订货号 (MLFB)	6ES7 290-6AA30-0XA0
电缆长度	2 m
重量	200 g

I/O 扩展电缆有一个公连接器和一个母连接器。

1. 将公连接器连接到信号模块右侧的总线连接器。
2. 将母连接器连接到信号模块左侧的总线连接器。
   - 将母连接器的钩伸端滑入总线连接器处的外壳
   - 将母连接器按入总线连接器中。

B

计算功率预算

CPU 有一个内部电源，用于为 CPU 本身和任何扩展模块供电以及满足其它 24 VDC 用户的功率要求。

有三种类型的扩展模块：

- 信号模块 (SM) 安装在 CPU 右侧。 在不考虑功率预算的情况下，每个 CPU 可允许的最大信号模块数如下。
  - CPU 1214 允许 8 个信号模块
  - CPU 1212 允许 2 个信号模块
  - CPU 1211 不允许任何信号模块
- 通信模块 (CM) 安装在 CPU 左侧。 若不考虑功率预算，任何 CPU 都允许最多 3 个通信模块。
- 信号板 (SB) 安装在 CPU 顶部。 任何 CPU 都允许最多 1 个信号板。

请使用以下信息作为指导，确定 CPU 可为您的组态提供多少电能（或电流）。

每个 CPU 都提供了 5 VDC 和 24 VDC 电源：

- 连接了扩展模块时，CPU 会为这些扩展模块提供 5 VDC 电源。 如果扩展模块的 5 VDC 功率要求超出 CPU 的功率预算，则必须拆下一些扩展模块直到其功率要求在功率预算范围内。
- 每个 CPU 都有一个 24 VDC 传感器电源，该电源可以为本地输入点或扩展模块上的继电器线圈供给 24 VDC。 如果 24 VDC 的功率要求超出 CPU 的功率预算，则可以增加外部 24 VDC 电源为扩展模块供应 24 VDC。 必须将 24 VDC 电源手动连接到输入点或继电器线圈。

⚠警告

将外部 24 VDC 电源与 DC 传感器电源并联会导致这两个电源之间有冲突，因为每个电源都试图建立自己首选的输出电压电平。

该冲突可能使其中一个电源或两个电源的寿命缩短或立即出现故障，从而导致 PLC 系统的运行不确定。 运行不确定可能导致死亡、人员重伤和/或财产损失。

CPU 上的 DC 传感器电源和任何外部电源应分别给不同位置供电。 允许将多个公共端连接到一个位置。

PLC 系统中的一些 24 V 电源输入端口是互连的，并且通过一个公共逻辑电路连接多个 M 端子。 在数据表中指定为非隔离时，CPU 的 24 VDC 电源输入、SM 继电器线圈电源输入以及非隔离模拟电源输入即是一些互连电路的实例。 所有非隔离的 M 端子必须连接到同一个外部参考电位。

⚠警告

将非隔离的 M 端子连接到不同参考电位将导致意外的电流，该电流可能导致 PLC 和连接设备损坏或运行不确定。
这种损坏或不确定运行可能导致死亡、人员重伤和/或财产损失。
务必确保 PLC 系统中的所有非隔离 M 端子都连接到同一个参考电位。

有关 CPU 功率预算和信号模块功率要求的信息，请参见技术规范 (页 321)。

说明

若超出 CPU 功率预算，将导致无法连接 CPU 所允许的最大数量的模块。

B.2 功率要求计算实例

以下实例是 PLC 功率要求的一个计算实例，该 PLC 包括一个 CPU 1214C AC/DC/继电器型、3 x SM 1223 8 DC 输入/8 继电器输出和 1 x SM 1221 8 DC 输入。该实例一共有 46 点输入和 34 点输出。

说明

该 CPU 已分配驱动内部继电器线圈所需的功率。 功率预算计算中无需包括内部继电器线圈的功率要求。

在本例中的 CPU 为 SM 提供了足够的 5 VDC 电流，但没有通过传感器电源为所有输入和扩展继电器线圈提供足够的 24 VDC 电流。 I/O 需要 448 mA 而 CPU 只提供 400 mA。 该安装额外需要一个至少为 48 mA 的 24 VDC 电源以运行所有包括的 24 VDC 输入和输出。

CPU 功率预算	**5 VDC**	**24 VDC**
CPU 1214C AC/DC/继电器	1600 mA	400 mA
减		
系统要求	**5 VDC**	**24 VDC**
CPU 1214C，14 点输入	-	14 * 4 mA = 56 mA
3 个 SM 1223，5 V 电源	3 * 145 mA = 435 mA	-
1 个 SM 1221，5 V 电源	1 * 105 mA = 105 mA	-
3 个 SM 1223，各 8 点输入	-	3 * 8 * 4 mA = 96 mA
3 个 SM 1223，各 8 个继电器线圈	-	3 * 8 * 11 mA = 264 mA
1 个 SM 1221，8 点输入	-	8 * 4 mA = 32 mA
总要求	540 mA	448 mA
等于		
电流差额	**5 VDC**	**24 VDC**
总电流差额	1060 mA	(48 mA)

B.3 计算功率要求

通过下表可以确定 S7-1200 CPU 可为您的组态提供多少电源（或电流）。 有关用户 CPU 型号的功率预算和信号模块功率要求信息，请参见技术规范 (页 321)。

CPU 功率预算	5 VDC	24 VDC
减		
系统要求	**5 VDC**	**24 VDC**
总要求		
等于		
电流差额	**5 VDC**	**24 VDC**
总电流差额		

C

订货号

CPU		订货号
CPU 1211C	CPU 1211C DC/DC/DC	6ES7 211-1AD30-0XB0
	CPU 1211C AC/DC/继电器	6ES7 211-1BD30-0XB0
	CPU 1211C DC/DC/继电器	6ES7 211-1HD30-0XB0
CPU 1212C	CPU 1212C DC/DC/DC	6ES7 212-1AD30-0XB0
	CPU 1212C AC/DC/继电器	6ES7 212-1BD30-0XB0
	CPU 1212C DC/DC/继电器	6ES7 212-1HD30-0XB0
CPU 1214C	CPU 1214C DC/DC/DC	6ES7 214-1AE30-0XB0
	CPU 1214C AC/DC/继电器	6ES7 214-1BE30-0XB0
	CPU 1214C DC/DC/继电器	6ES7 214-1HE30-0XB0

信号模块、通信模块和信号板		订货号
信号模块	SM 1221 8 x 24 VDC 输入	6ES7 221-1BF30-0XB0
	SM 1221 16 x 24 VDC 输入	6ES7 221-1BH30-0XB0
	SM 1222 8 x 24 VDC 输出	6ES7 222-1BF30-0XB0
	SM 1222 16 x 24 VDC 输出	6ES7 222-1BH30-0XB0
	SM 1222 8 x 继电器输出	6ES7 222-1HF30-0XB0
	SM 1222 16 x 继电器输出	6ES7 222-1HH30-0XB0
	SM 1223 8 x 24 VDC 输入/8 x 24 VDC 输出	6ES7 223-1BH30-0XB0
	SM 1223 16 x 24 VDC 输入/16 x 24 VDC 输出	6ES7 223-1BL30-0XB0
	SM 1223 8 x 24 VDC 输入/8 x 继电器输出	6ES7 223-1PH30-0XB0
	SM 1223 16 x 24 VDC 输入/16 x 继电器输出	6ES7 223-1PL30-0XB0
	SM 1231 4 x 模拟量输入	6ES7 231-4HD30-0XB0
	SM 1231 8 x 模拟量输入	6ES7 231-4HF30-0XB0
	SM 1232 2 x 模拟量输出	6ES7 232-4HB30-0XB0

信号模块、通信模块和信号板		订货号
	SM 1232 4 x 模拟量输出	6ES7 232-4HD30-0XB0
	SM 1234 4 x 模拟量输入/2 x 模拟量输出	6ES7 234-4HE30-0XB0
通信模块	CM 1241 RS232	6ES7 241-1AH30-0XB0
	CM 1241 RS485	6ES7 241-1CH30-0XB0
信号板	SB 1223 2 x 24 VDC 输入/2 x 24 VDC 输出	6ES7 223-0BD30-0XB0
	SB 1232 1 路模拟量输出	6ES7 232-4HA30-0XB0

HMI 设备	订货号
KTP400 Basic（单色，PN）	6AV6 647-0AA11-3AX0
KTP600 Basic（单色，PN）	6AV6 647-0AB11-3AX0
KTP600 Basic（彩色，PN）	6AV6 647-0AD11-3AX0
KTP1000 Basic（彩色，PN）	6AV6 647-0AF11-3AX0
TP1500 Basic（彩色，PN）	6AV6 647-0AG11-3AX0

编程数据包	订货号
STEP 7 Basic v10.5	6ES7 822-0AA0-0YA0

存储卡、其它硬件和备件		订货号
存储卡	SIMATIC MC 2 MB	6ES7 954-8LB00-0AA0
	SIMATIC MC 24 MB	6ES7 954-8LF00-0AA0
其它硬件	PSU 1200 电源	6EP1 332-1SH71
	CSM 1277 以太网交换机 - 4 端口	6GK7 277-1AA00-0AA0
	仿真器（1214C/1211C - 8 位置）	6ES7 274-1XF30-0XA0
	仿真器（1214C - 14 位置）	6ES7 274-1XH30-0XA0
	I/O 扩展电缆，2 m	6ES7 290-6AA30-0XA0
备件	连接器板，7 个端子，镀锡	6ES7 292-1AG30-0XA0
	连接器板，8 个端子，镀锡 (4/pk)	6ES7 292-1AH30-0XA0

存储卡、其它硬件和备件		订货号
	连接器板，11 个端子，镀锡 (4/pk)	6ES7 292-1AL30-0XA0
	连接器板，12 个端子，镀锡 (4/pk)	6ES7 292-1AM30-0XA0
	连接器板，14 个端子，镀锡 (4/pk)	6ES7 292-1AP30-0XA0
	连接器板，20 个端子，镀锡 (4/pk)	6ES7 292-1AV30-0XA0
	连接器板，3 个端子，镀金 (4/pk)	6ES7 292-1BC0-0XA0
	连接器板，6 个端子，镀金 (4/pk)	6ES7 292-1BF30-0XA0
	连接器板，7 个端子，镀金 (4/pk)	6ES7 292-1BG30-0XA0
	连接器板，11 个端子，镀金 (4/pk)	6ES7 292-1BL30-0XA0

文档	订货号
S7-1200 可编程控制器系统手册	
• 德语	• 6ES7 298-8FA30-8AH0
• 英语	• 6ES7 298-8FA30-8BH0
• 法语	• 6ES7 298-8FA30-8CH0
• 西班牙语	• 6ES7 298-8FA30-8DH0
• 意大利语	• 6ES7 298-8FA30-8EH0
• 中文	• 6ES7 298-8FA30-8KH0
S7-1200 简明手册	
• 德语	• 6ES7 298-8FA30-8AQ0
• 英语	• 6ES7 298-8FA30-8BQ0
• 法语	• 6ES7 298-8FA30-8CQ0
• 西班牙语	• 6ES7 298-8FA30-8DQ0
• 意大利语	• 6ES7 298-8FA30-8EQ0
• 中文	• 6ES7 298-8FA30-8KQ0

索引

T

U

V

X

上

下

不

专

丢

中

串

主

乘

事

交

扫

技

指

排

接

插

继

编

网

置

联

背

脉

自

范

规

选

通

配

重

错

长

门

除

项

额

高