Simatic S7 到 Logix5000 应用程序转换指南

应用解决方案

Allen-Bradley • Rockwell Software
Rockwell Automation

重要用户信息

固态设备的工作特性与机电设备不同。《固态控制器的应用、安装和维护安全指南》（出版号 SGI-1.1，可向当地的罗克韦尔自动化销售处索取，也可通过 http://literature.rockwellautomation.com 在线访问）介绍了固态设备与硬接线机电设备之间的一些重要区别。由于这些区别以及固态设备的广泛应用，负责应用此设备的所有人员都必须确保对此设备的所有应用均符合要求。

在任何情况下，对于因使用或应用此设备而导致的任何间接或连带损失，罗克韦尔自动化概不负责。

本手册中的示例和图表仅供说明之用。由于具体安装中存在许多可变因素和不同要求，所以罗克韦尔自动化对于按照示例或图表进行的实际应用不承担任何责任或义务。

对于本手册中所述信息、电路、设备或软件的使用，罗克韦尔自动化不承担任何专利责任。

未经罗克韦尔自动化的书面许可，不得复制本手册的全部或部分内容。

在整本手册中，我们在必要的地方做出了说明，以告知您安全注意事项。

	标识有关在危险环境下可能导致爆炸进而可能造成人身伤害或死亡、财产损坏或经济损失的行为或情况的信息。
	标识对成功应用和认识产品至关重要的信息。
	标识有关可能造成人身伤害或死亡、财产损坏或经济损失的行为或情况的信息。注意事项有助于您辨别危险、避免危险和认识后果
	此类标签可能贴在设备表面，也可能贴在设备内部 （如驱动器或电机），用于警告可能存在危险电压。
	此类标签可能贴在设备表面，也可能贴在设备内部 （如驱动器或电机），用于警告可能存在高温危险。

目录

章 3

系统软件和标准功能的转换

章 4

典型程序结构的转换

章 5

转换到 Logix 时的常见错误

目标

本用户手册为使用基于以下两种平台之一的控制系统的用户和工程师提供指导：

- Siemens S7 控制器
- 罗克韦尔自动化 Logix 可编程自动化控制器 (PAC)

此外，本指南还面向：

- 希望或需要利用 PAC 功能，或者已经处于从 S7 迁移到 Logix 的早期阶段的用户。
- 希望将特定的 STEP 7 程序代码转换为高效的 RSLogix 5000 代码的用户。

使用本指南可帮助您在将项目转换为 Logix 时采用最佳做法并避免常见错误。

转换与转译

转换与转译是本应用程序转换指南中反复出现的主题。简单转译仅仅是针对代码行并找到 Logix 语言的等效代码。若要对应用程序进行最佳转换，不能只局限于转译层面。例如，如果选择不同的编程语言、利用不同的编程技术、设计不同的调度方案来解决相同的任务，您可能会从中获益。因此，转换是在了解 Logix 系统强大功能的前提下进行更高级设计时执行的。

如果要转换应用程序代码，那么在开始转换之前，需要了解 STEP 7 程序 - 为此，您可以亲自参与该程序的开发，也可以阅读有关该程序及其所控制的过程的文档。如果不熟悉程序或过程，或者其文档不完善，将难以执行正确的转换 ？ 这样只能执行转译，而且很可能不会成功。例如，在 Logix 中，有一个全局命名空间，而在 Siemens 环境中，有多个可通过应用程序代码加载 / 卸载的数据块。了解这些有助于设计转换策略。

在某些情况下，如果过程和程序的文档都不完善，那么制定一个新规范，以最短的时间转译旧程序，然后在此基础上开始开发新的 Logix 程序，则可能降低项目的总持续时间 / 总成本。

术语

STEP 7 是 Siemens SIMATIC S7 控制器的编程软件环境。RSLogix 5000 软件用于罗克韦尔自动化 Logix 可编程自动化控制器。我们之所以将 Logix 称为可编程自动化控制器，是因为它的功能比传统的通用 PLC 更加强大。它提供了一个用于多分类控制的优秀控制平台、一个公共命名空间、用于真正可伸缩的 CPU 体系结构的协调系统时间、用户定义数据类型以及完全的 NetLinx 连接。

术语 “Logix”用于指 ControlLogix、CompactLogix、GuardLogix、FlexLogix、DriveLogix 或 SoftLogix 控制器的任何一种，也指 RSLogix 5000 编程环境，具体可根据其引用上下文确定。

其他资源

本应用程序转换指南的每一节都引用了其他罗克韦尔自动化用户手册、选型指南和文档，在这些资料中可找到更多信息。

出版号	出版物标题
1756-SG001	ControlLogix 控制器选型指南
1769-SG001	1769 CompactLogix 控制器选型指南
1768-UM001	1768 CompactLogix 控制器用户手册
1769-SG002	Compact I/O 选型指南
1756-RM094	Logix5000 Controllers Design Considerations Programming Manual （Logix5000 控制器设计要点编程手册）
1756-PM001	Logix5000 Controllers Common Procedures Programming Manual （Logix5000 控制器通用编程步骤手册）
1756-RM003	Logix5000 Controllers General Instructions Reference Manual （Logix5000 控制器通用指令参考手册）
1734-SG001	POINT I/O 选型指南
1738-SG001A-ZH-P	ArmorPoint I/O 选型指南
1792-SG001	ArmorBlock MaXum I/O 和 ArmorBlock I/O 选型指南
1794-SG002	FLEX I/O 和 FLEX Ex 选型指南
NETS-SG001	NetLinx 选型指南
VIEW-SG001	可视化平台选型指南
IA-RM001	Integrated Architecture: Foundations of Modular Programming （集成结构：模块化编程基础）
6873-SG004	Encompass Program Product Directory （Encompass 项目的产品目录）

出版号	出版物标题
1756-PM010	Logix5000 Controllers Add-On Instructions Programming Manual （Logix5000 控制器附加指令编程手册）
1756-RM087	Logix5000 控制器执行时间和内存使用情况参考手册
IASIMP-RM001	IA Recommended Literature Reference Manual （IA 推荐资料参考手册）

罗克韦尔自动化提供的 PLC 逻辑转换服务

罗克韦尔自动化提供额外的 PLC 逻辑转换服务。

- 服务特色
- 一站式 PLC 程序转换服务
- 服务优势
- 提供的服务
- 基本转换服务包
- 转换加上初始清理包
- 其他程序转换

服务特色

程序转换服务旨在转换传统的 Allen-Bradley 牌 PLC 或第三方可编程控制器程序，以便在 Logix 可编程自动化控制系统或 SLC 500/MicroLogix 或 PLC-5 可编程控制器上运行。

传统产品通常需要较高的支持费用且难以维修，从而导致停机时间增加，生产率降低。为此，罗克韦尔自动化客户支持部门现在提供程序转换服务。这类服务旨在降低从传统 PLC 迁移到本公司最新的 PAC 或 PLC 控制平台系列产品之一所需的成本和时间。

一站式 PLC 程序转换服务

从传统产品迁移到最新的 Allen-Bradley 控制平台将改善制造过程、提高系统的可靠性和灵活性、提供更多应用程序处理功能，并降低设备维修成本和备件库存。借助罗克韦尔自动化客户支持部门的程序转换服务，可将现有的可编程控制器程序专业快速地转换至新的控制器产品系列。罗克韦尔自动化客户支持工程师可帮助您迁移传统 Allen-Bradley 设备，也可将 PLC 系统转换到罗克韦尔自动化产品，同时最大限度降低停机时间并提高操作成功率。

服务优势

每种产品平台的专业人员都将参与程序转换过程。查找因输入错误而引起的逻辑异常并非难事。大多数情况下将重建整个数据表，并且不丢失任何数据，同时保留原始文档，不需要重新输入注释和符号。原始的 Allen-Bradley 程序可能采用 6200、APS 或 AI 系列格式。新程序将采用适当的 RSLogix 格式。

提供的服务

提供两种程序转换服务包，同时针对个案提供特定于项目的定制服务包。

基本转换服务包

- 原始的可编程控制器程序将转换为相应的 ControlLogix、CompactLogix、PLC-5 或 SLC 500/MicroLogix 格式。
- 该服务包提供在转换过程中生成的错误清单，其中包括无法直接转换的指令和所有可能未转换的地址，这些地址可能包括指针和间接寻址。
- 程序和错误清单将返回给客户，以便进行手动调试和更正。

转换加上初始清理包

- 原始的可编程控制器程序将转换为相应的 ControlLogix、PLC-5 或 SLC 500/MicroLogix 格式。
- 我们将更正所有指令和 / 或寻址错误，并转换到新的处理器产品系列。
- 转换后的程序将返回给客户，以便进行最终启动和调试。

附加选项

上述两种服务包的附加选项包括：

- 项目启动和调试阶段的应用程序级电话支持。
- 有关系统重建、操作员接口、体系结构和通讯策略的咨询（以便充分利用新平台中不属于代码转译工作的控制功能）以及培训和现场启动均作为增值服务，由当地的全球销售和解决方案 (GSS) 部门提供。
- 当地的 GSS/ 工程系统部门可提供全包的迁移或升级服务。

其他程序转换

- PLC-2 格式转换为 ControlLogix、CompactLogix、PLC-5 和 SLC500/MicroLogix 格式
- PLC-3 格式转换为 ControlLogix、CompactLogix 或 PLC-5 格式
- PLC-5/250 格式转换为 ControlLogix 或 CompactLogix 格式
- Modicon - Quantum、984、584、380、381、480、485、780、785 转换为 ControlLogix 或 CompactLogix 格式
- Siemens - S-5、S-7 转换为 ControlLogix 或 CompactLogix 格式
- TI - 520、520C、525、530、530C、535、560、560/565、565、560/560T、560T、545、555、575 转换为 ControlLogix 或 CompactLogix 格式
- GE Series 6 转换为 ControlLogix 或 CompactLogix 格式

此外，还可以提供从其他第三方可编程控制器到 Allen-Bradley 控制器程序的程序转换。有关详细信息，请联系技术支持部门。

如需安排转换项目，或者想了解程序转换服务的更多信息，请联系当地的罗克韦尔自动化销售处或授权经销商：请发电子邮件至 raprogramconversion@ra.rockwell.com，或者访问 http://support.rockwellautomation.com/ 并查阅知识库文档 G19154。

重要事项 利用重建咨询服务通常是为了扩展系统功能，而不是因为陈旧或相关原因而更换硬件。RSLogix 5000 软件内置有 SLC 到 Logix 格式的转换、PLC-5 到 Logix 格式的转换和 PCE 注释生成功能。

注：

硬件转换

简介

本章旨在为需要确定哪种 Logix 硬件适合于替换现有 S7 设备的用户或工程师提供相关指导。

本章介绍如何选择控制器、本地 I/O、远程 I/O、网络和 HMI，其中一节介绍分布式控制器结构。另外，本章还提供最常用的 S7 模块的硬件转换示例。

S7 控制器

下表列出了对应于当前的 Siemens S7 控制器的相关示例选型，这些控制器的应用范围非常广泛。

当前的 Siemens S7 控制器的示例选型

控制器	部件号	Logix 同等产品
313C	6ES7 313-5BF03-0AB0	L23 串行
314C-DP	6ES7 314-6CG03-0AB0	L23 EtherNet/IP、L31
315-2 DP	6ES7 315-2AG10-0AB0	L32E、L32C
317-2 DP	6ES7 317-6TJ10-0AB0	L35CR、L35E
317T-2 DP	6ES7 317-6TJ10-0AB0	L43、L45
319-3 PN/DP	6ES7 318-3EL00-0AB0	L45、L61
414-2	6ES7 414-2XK05-0AB0	L61、L62

当前的 Siemens S7 控制器的示例选型

414-3 414-3 PN/DP	6ES7 414-3XM05-0AB0 6ES7 414-3EM05-0AB0	L62、L63、L64、L65
315F-2 PN/DP （安全）	6ES7 315-2FH13-0AB0 6ES7 317-2FK13-0AB0	GuardLogix L61S、L62S、L63S
414-H （冗余） 417-H	6ES7 414-4HM14-0AB0 6ES7 417-4HT14-0AB0	L61-L65 （带 SRM）
PCS7 - 使用 417-4 控制器		L3x、L4x、L6x + FactoryTalk View、FactoryTalk Batch 软件

下面是部分最常用的 S7 控制器的适用性指南：

- S7 315-2DP – 中小型机器。
- S7 317-2DP – 中型到中大型机器，中小型过程控制应用。
- S7 414-2 – 要求严格的机器控制，过程控制应用。
- S7 414-3 – 要求严格的机器控制，大型过程控制应用。

附录 A 列出了全部 S7 控制器。

I/O 系统

下面各节介绍用来替换现有 S7 设备的 Logix I/O 系统。

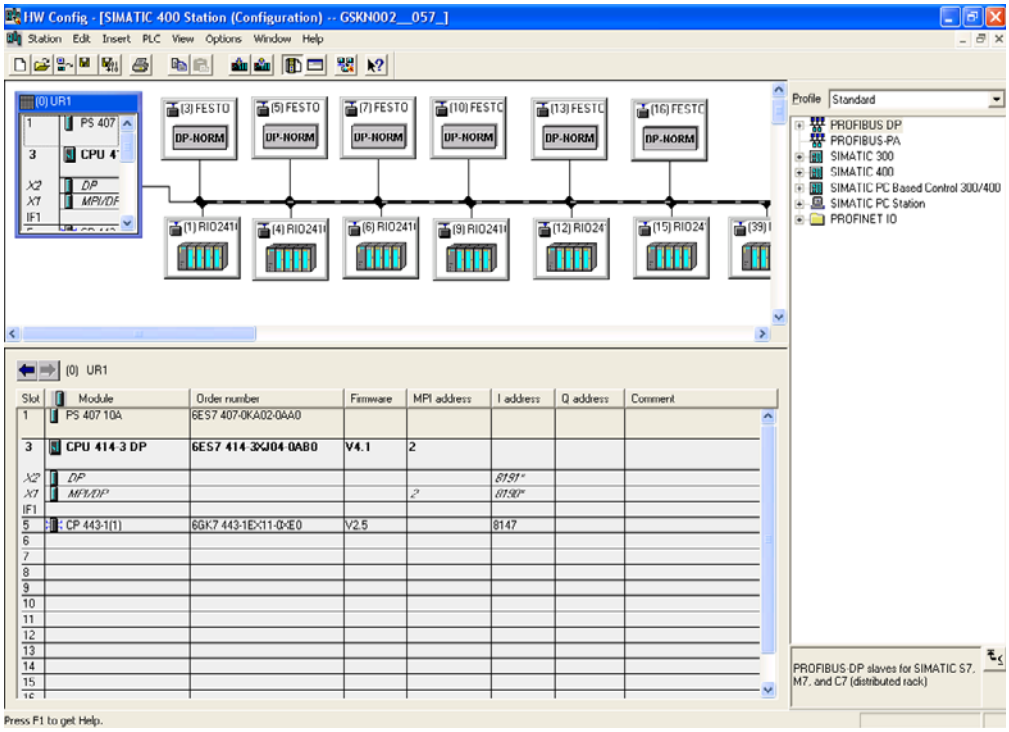
S7 本地 I/O

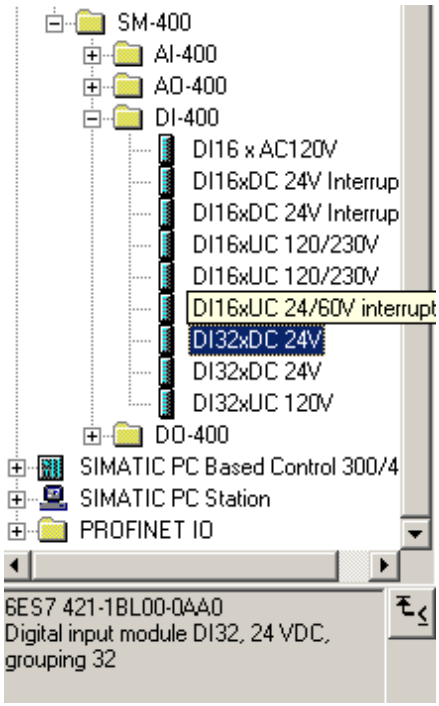
S7-300 和 S7-400 I/O 模块的种类繁多。S7-300 模块安装在标准 DIN 导轨上，并通过模块随附的 U 型连接器连接到相邻的卡上。S7-400 模块安装在 S7-400 机架上。

S7 I/O 组件的选型和配置

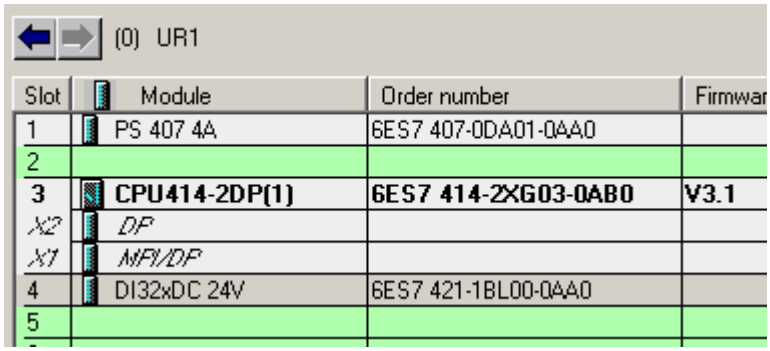
下面的屏幕截图来自 STEP 7 Hardware Configuration 程序，这是 STEP 7 应用程序集合中的一个独立程序。该功能已完全集成在 RSLogix 5000 软件中，本用户手册后面将进行介绍。

STEP 7 Hardware Configuration 程序





将选定的模块拖放到机架配置屏幕中。



Logix 本地 I/O

ControlLogix 和 CompactLogix I/O 模块的种类繁多。1769 I/O 具有优化成本的特点，可满足 OEM 通常对于功能够用即可的要求，而 1756 I/O 系列可为要求非常严格的应用提供高级功能，最终用户通常需要这些功能，有时，为了达到特定性能级别，也需要这些功能。

CompactLogix 模块安装在标准 DIN 导轨上，一个特殊的耦合系统确保与相邻模块的电气和机械连接。工程师可能愿意使用机械耦合系统 - 对于 S7-300，模块只固定在一个特殊导轨上，而不是相互连接（除非使用电气 U 型连接器）。

ControlLogix 模块安装在 1756 机架中。

- 对于 1769-L31、1769-L32C、1769-L32E 和 1768-L43 控制器，连接到控制器机架的 I/O 模块数最大为 16（最多在 3 个区块中）。
- 对于 1769-L35CR、1769-L35E 和 1768-L45 控制器，连接到控制器机架的 I/O 模块数最多为 30 个（也在 3 个区块中）。
- 对于 1756 控制器，机架中的插槽数决定了本地 I/O 模块的最大数量。插槽数可达 4、7、10、13 或 17 个。

在这两种平台上，均可通过 CIP 网络连接更多 I/O，其中 EtherNet/IP 和 ControlNet 网络可提供最紧密的无缝 I/O 集成。

下表列出了部分常用 S7 I/O 模块的 Logix 同等产品。

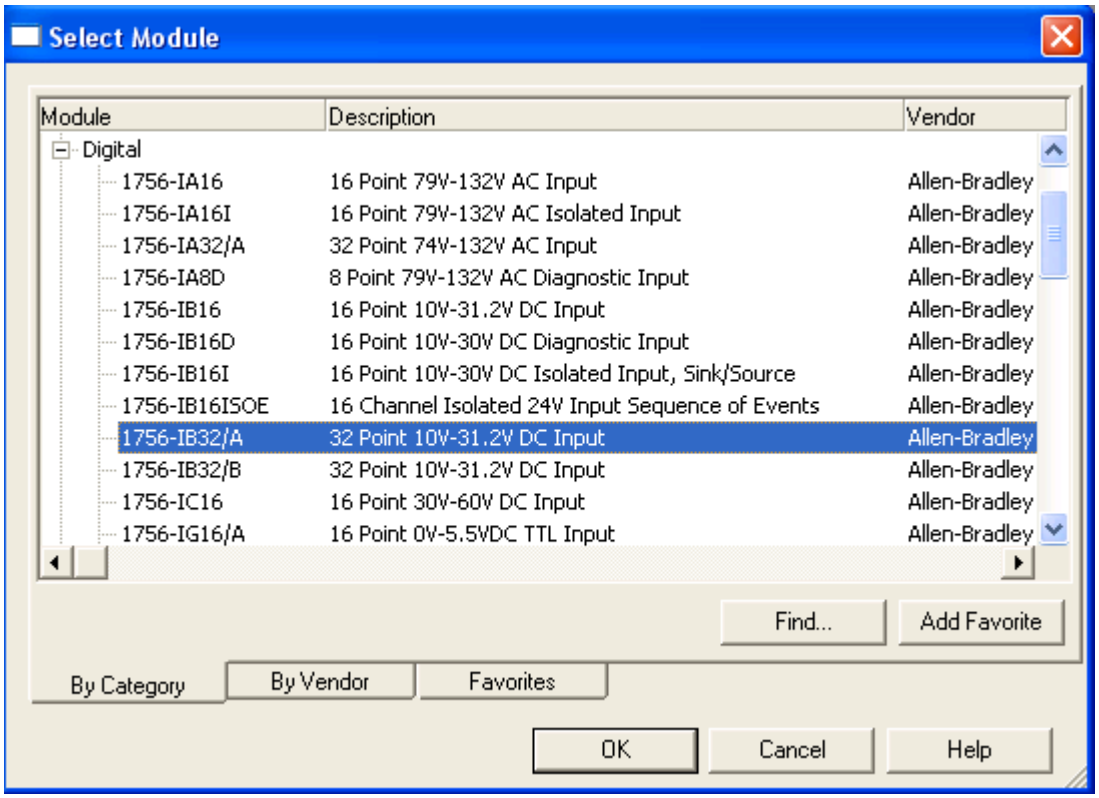
S7 I/O 模块的 Logix 同等产品

S7 I/O 模块	说明	Logix 同等产品	说明
6ES7 321-1BL00-0AA0	S7-300 32 通道数字输入	1769-IQ32	CompactLogix 32 通道数字输入
6ES7 322 - 1BH01-0AA0	S7-300 16 通道数字输出	1769-OB16	CompactLogix 16 通道数字输出
6ES7 421-1BL01-0AA0	S7-400 32 通道数字输入	1756-IB32	ControlLogix 32 通道数字输入
6ES7 422-1BH01-0AA0	S7-400 16 通道数字输出	1756-OB16E	ControlLogix 16 通道数字输出

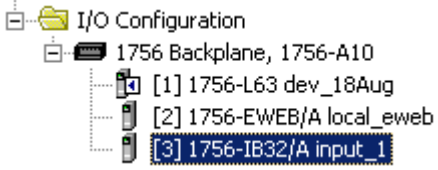
有关 I/O 模块的详细转换表，请参考附录 A。

Logix I/O 组件的选型和配置

从项目树的 I/O Configuration 分支中，可以访问设备配置文件的 Logix 库。这些配置文件提供完全向导式的配置，以便完全、方便地集成到数据表中，并对每个模块的功能 （如扩展、报警和诊断）提供直观的可编程控制。

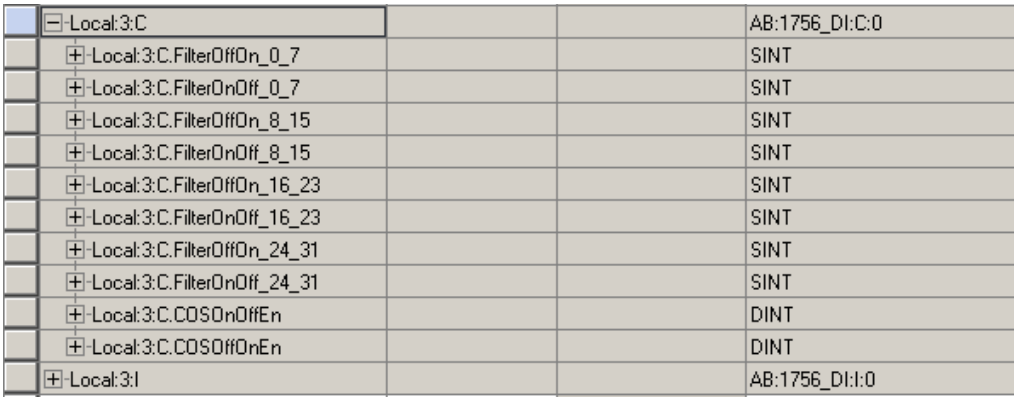


选择一个项后，该项将显示在 I/O 配置中的机架中。



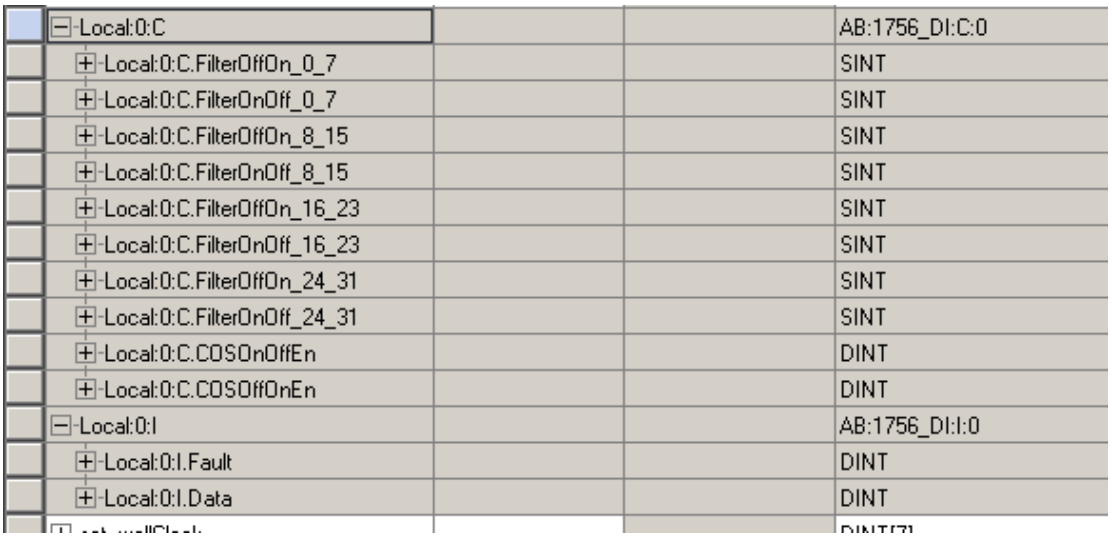
新 I/O 模块的设备配置文件标签已自动添加到控制器范围标签数据库。

下图显示部分展开的标签。



Local:3:C			AB:1756_DI:C:0
Local:3:C.FilterOffOn_0_7			SINT
Local:3:C.FilterOnOff_0_7			SINT
Local:3:C.FilterOffOn_8_15			SINT
Local:3:C.FilterOnOff_8_15			SINT
Local:3:C.FilterOffOn_16_23			SINT
Local:3:C.FilterOnOff_16_23			SINT
Local:3:C.FilterOffOn_24_31			SINT
Local:3:C.FilterOnOff_24_31			SINT
Local:3:C.COSOnOffEn			DINT
Local:3:C.COSOffOnEn			DINT
Local:3:I			AB:1756_DI:I:0

配置文件包含配置数据、状态数据以及 I/O 数据。



Local:0:C			AB:1756_DI:C:0
Local:0:C.FilterOffOn_0_7			SINT
Local:0:C.FilterOnOff_0_7			SINT
Local:0:C.FilterOffOn_8_15			SINT
Local:0:C.FilterOnOff_8_15			SINT
Local:0:C.FilterOffOn_16_23			SINT
Local:0:C.FilterOnOff_16_23			SINT
Local:0:C.FilterOffOn_24_31			SINT
Local:0:C.FilterOnOff_24_31			SINT
Local:0:C.COSOnOffEn			DINT
Local:0:C.COSOffOnEn			DINT
Local:0:I			AB:1756_DI:I:0
Local:0:I.Fault			DINT
Local:0:I.Data			DINT

有关更多信息，请参考第 4 章。

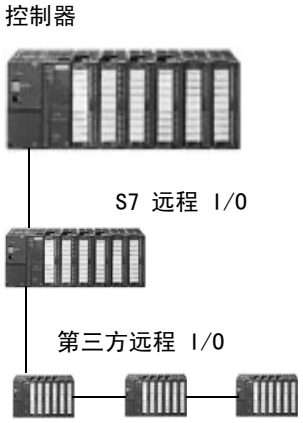
S7 远程 I/O

常见的做法是，在控制器的本地机架与远程 I/O 工作站之间将 I/O 分开，通过 Profibus DP 网络进行通讯。下面是 Profibus DP 节点的类型：

- S7 远程 I/O，其中标准 S7-300 I/O 模块安装在远程 I/O 面板上，通过一个特殊的模块与 Profibus DP 总线连接。控制器将此 I/O 视作本地 I/O，并分配标准 I/O 地址。这称为 ET200M。

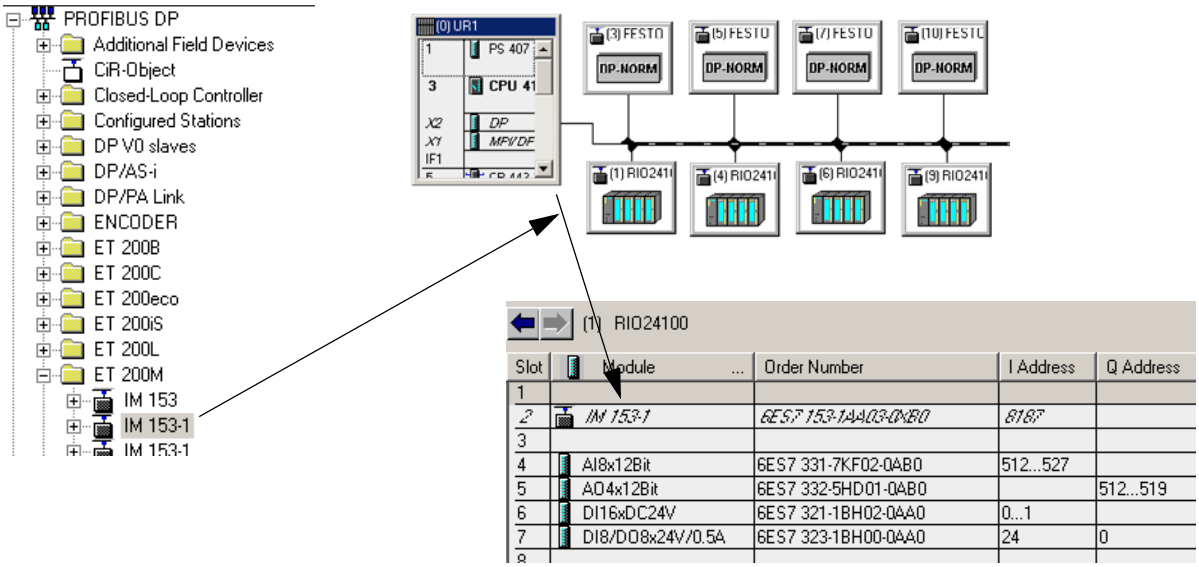
- 其他 Siemens 远程 I/O，如 ET200S （类似于 POINT I/O 系统）和 ET200L （类似于 FLEX I/O 系统）。
- 第三方远程 I/O。很多 I/O 和阀门制造商都生产接口，采用 S7 远程 I/O 那样的方式，将其系统连接到 Profibus DP 总线。对于这些系统，可能需要将一个特殊的集成文件 （GSD 文件）导入 STEP 7 安装。
- 某些生产更复杂设备 （如称重仪和变速驱动器 (VSD)）的制造商为其产品生产 Profibus DP 接口。对于这些系统，需要将一个特殊的集成文件 （GSD 文件）导入 STEP 7 安装。这通常需要参考制造商的文档，以了解数据区的含义。

典型 S7 I/O 配置



S7 Profibus DP 远程 I/O 的配置

将 Profibus DP 接口模块从硬件目录拖放到 Profibus DP 总线图上，即可将其安装在硬件配置中。安装接口模块后，可以将其打开，将标准 S7-300 模块添加到其中，就好像它是本地 I/O 一样。



数据表定义与驱动器关联的 I/O 地址。这些地址的符号将手动添加到符号表中。现在完成了硬件配置。

Profibus DP 网络上的远程设备可以与 Logix 一起使用，但会受到与 S7 环境中相同的约束 / 可用性限制。

Logix 分布式 I/O

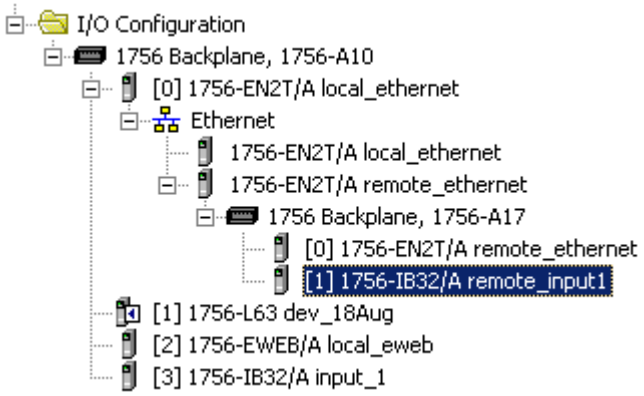
罗克韦尔自动化分布式 I/O 包括使用 1756 或 1769 I/O 模块的远程 I/O 和各种分布式 I/O 平台，如 POINT I/O、FLEX I/O、ArmorPoint 和 ArmorBlock 系统。

这些 I/O 模块通过通讯模块或通讯适配器连接到网络，或者使用内置通讯接口直接连接到网络。

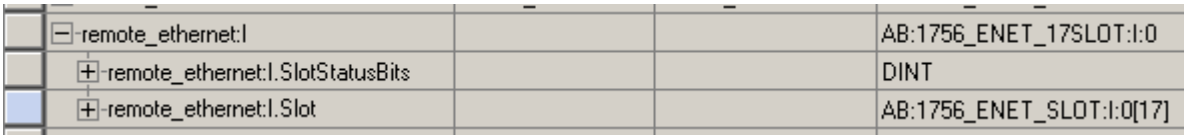
Logix 分布式 I/O 的配置

所有 I/O 配置都是在 RSLogix 5000 软件的项目树中完成的。从 I/O Configuration 分支中，为所选网络类型插入通讯模块。

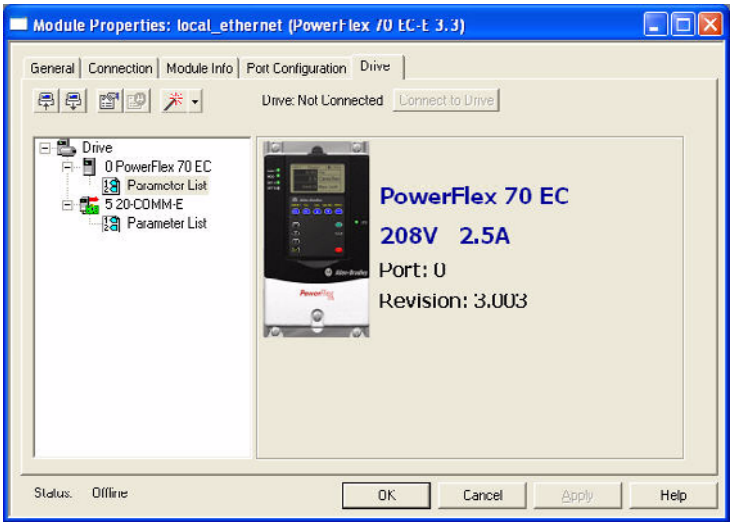
下面的屏幕截图显示新增了一个通过 EtherNet/IP 网络连接的远程 1756-IB32 I/O 模块。



注意，对应于该远程 I/O 模块的标签已自动添加到控制器作用域标签数据库中。



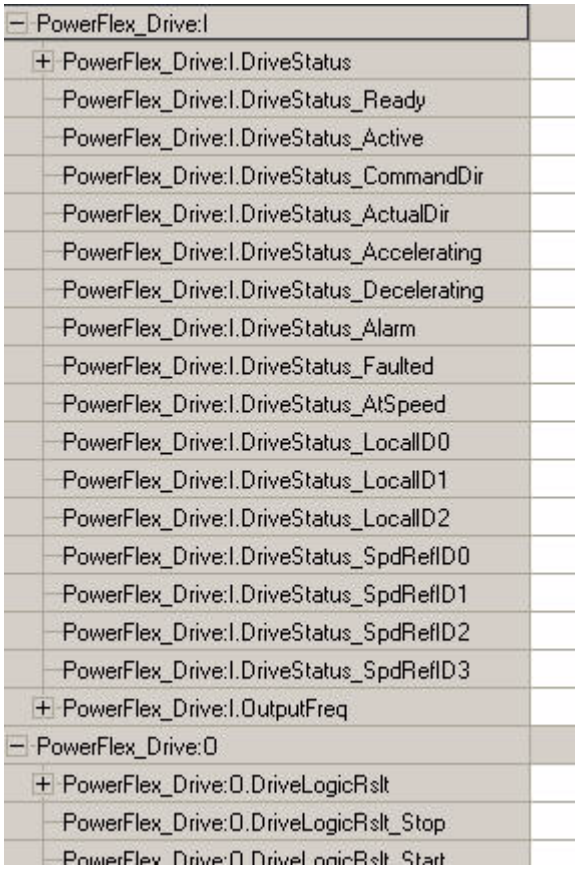
可通过同样的方法添加网络变速驱动器，如 PowerFlex 驱动器。



同样，对于在 RSLogix 5000 软件中具有配置文件且通过 EtherNet/IP 或 ControlNet 网络连接的所有设备，RSLogix 5000 软件都将自动为其生成新标签。对于 DeviceNet 网络，GuardLogix Safety I/O 以相同的方式集成。其他 DeviceNet 设备需要使用实际

功能等效于 STEP 7 Profibus 管理器软件和 GSD 文件的 RSNetWorx 配置软件和 EDS 文件进行设置。

下图显示的是 RSLogix 5000 软件中的设备配置文件标签，成百上千种罗克韦尔自动化设备都有此标签。



网络

有关网络的信息，请参考以下各节。

S7 中的网络

Profibus DP 网络、DPV1、DPV3

在 S7 环境下，设备的主要通讯网络类型是以各种形式实施的 Profibus DP 网络。部分高级 S7-300 和所有 S7-400 控制器都有内置的 Profibus 主端口。

Profibus 网络 - 其他

Profibus FMS 和 FDL 用于在控制器之间进行数据通讯。它们的功能与工业以太网相似，配置也几乎相同。区别在于 Profibus FMS 和 FDL 需要 Profibus 通讯处理器，而以太网不需要，此外前者还将采用 Profibus 布线。

在 S7-315T 和 S7-317T 控制器中，可以使用 Profibus DPv2 连接到伺服驱动器，用于低端运动控制。

工业以太网

Siemens 工业以太网是工业环境中的一种 Siemens 以太网。它主要用于控制器之间的通讯和控制器与编程计算机的通讯。

除了为 Profinet 配备的部分最新控制器外，S7 控制器没有内置的以太网端口。使用工业以太网的 S7 系统需在机架中安装通讯处理器。

根据通讯处理器，可使用以下协议：

- S7 （S7 控制器之间的专属通讯协议）
- TCP （传输控制协议）原始套接字
- ISO-on-TCP （带附加校验的扩展 TCP）
- UDP （用户数据报协议）原始套接字

这些网络上的大部分通讯管理问题，都需要有应用程序代码。

在罗克韦尔自动化环境中，可以使用集成的 EtherNet/IP 端口、EtherNet/IP 桥接模块和 / 或 EWEB 模块来实现此功能。

Profinet

Profinet 以相同的编程开销要求，在工业以太网上提供了相似的 Profibus DP 功能。使用 Profinet 的网络类似于 Profibus，区别是电缆和连接器不同，并且 Profinet 使用以太网现场接口模块，而 Profibus 不使用。带有内置 Profinet 接口或者专为 Profinet 而配备的通讯处理器的控制器用于连接到网络。

此外，现有 Profibus DP 网络可通过代理或使用配备了 Profinet 的控制器的 Profibus DP 端口桥接到 Profinet。

部分 Profinet 现场接口模块的集成开关有多个 RJ45 端口，必要时可实现 Profibus 类型的线路总线拓扑。

Profinet 提供以下三种通讯方式：

- Profinet CBA （基于组件的自动化），它主要用于控制器到控制器的通讯，使用标准以太网硬件和 TCP/IP 软件栈。
- 用于有调度传输 （如驱动器或 I/O 模块）的 Profinet IO，使用标准以太网硬件，但不使用 TCP/IP 软件栈。
- 用于运动控制应用的 Profinet IRT （同步实时），使用特定于 Profinet 的硬件，同样不使用 TCP/IP 软件栈，必须存在于受保护的网段中。

如果使用 Profinet CBA 框架，则可以通过图形配置集成 Profibus、Profinet 和工业以太网，而无需额外的编程。罗克韦尔自动化 EtherNet/IP 网络通过使用标准硬件和标准 TCP/IP 软件栈，并利用 Message (MSG) 指令和生产 / 消费标签之类的内置功能提供此功能。

Logix 中的网络

NetLinx 这一术语指网络技术领域中的罗克韦尔自动化解决方案。下面是 Logix 系统中主要使用的网络：

- EtherNet/IP
- ControlNet
- DeviceNet

这些网络具有很多重要的功能。它们都按照通用工业协议 (CIP) 设计，可用于控制、配置和收集任何 NetLinx 网络上的数据。因此，数据可以在不同网络之间传输，完全不需要协议转换软件或代理。

正在熟悉 Logix 系统的工程师可能会对 Logix 网络配置中的集成性和精巧性留下深刻印象。

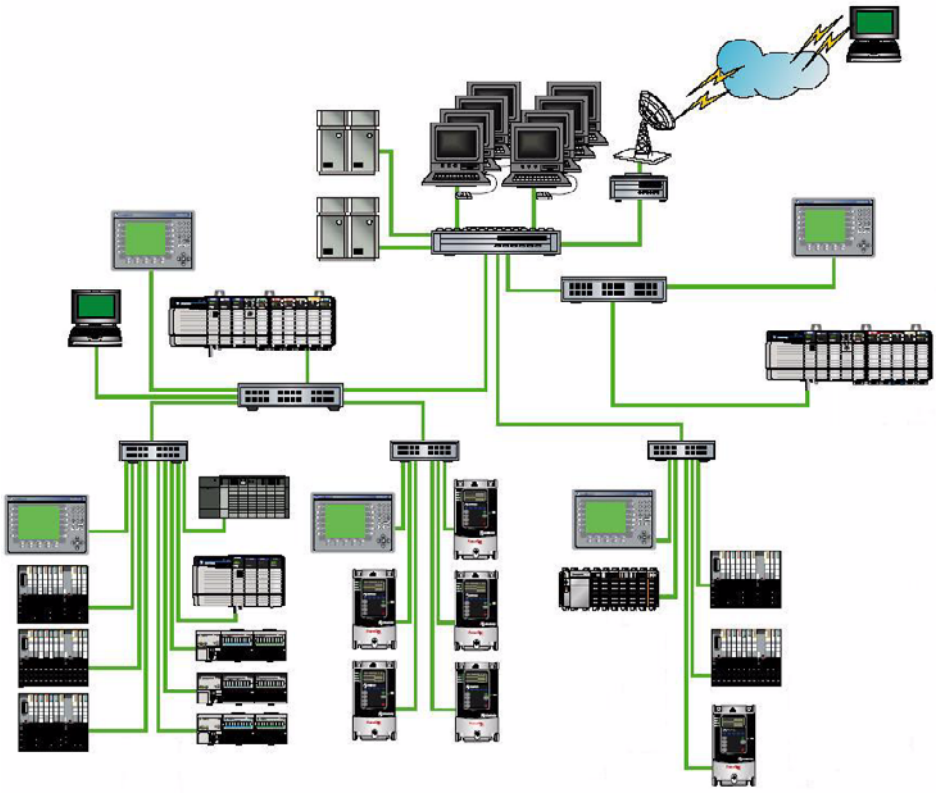
EtherNet/IP 网络

EtherNet/IP 网络提供全套的控制、配置和数据收集服务。它使用 TCP/IP 进行常规消息传递 / 信息交换，使用 UDP/IP 进行 I/O 消息传递。它最常用在以下类型的配置中：

- 常规 I/O 控制
- 控制器之间的数据交换
- 连接大量计算机
- 连接大量设备
- 与企业系统的连接

- 安全设备的集成
- 运动控制 （将来）

典型的 Ethernet/IP 示例

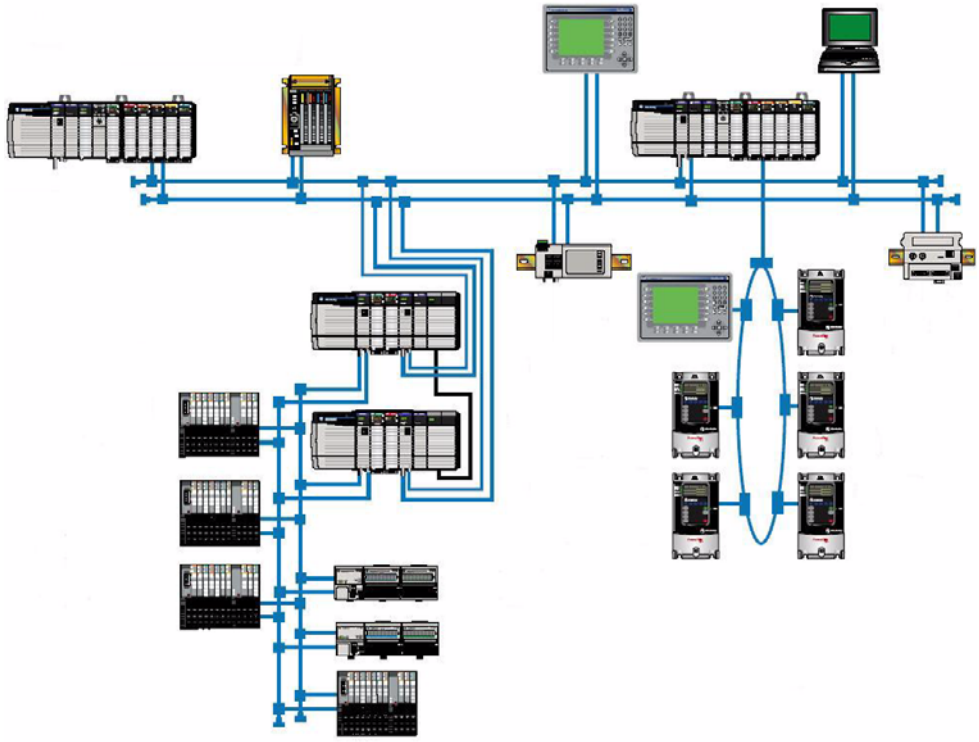


ControlNet 网络

ControlNet 是一种实时控制网络，用于传输对时间要求严格的 I/O 以及互锁数据和消息传递数据，包括在单条物理介质链路上上载 / 下载编程和配置数据。它最常用在以下类型的配置中：

- 常规 I/O 控制
- 控制器之间的数据交换
- 多个分布式 DeviceNet 网络的主干

典型的 ControlNet 示例

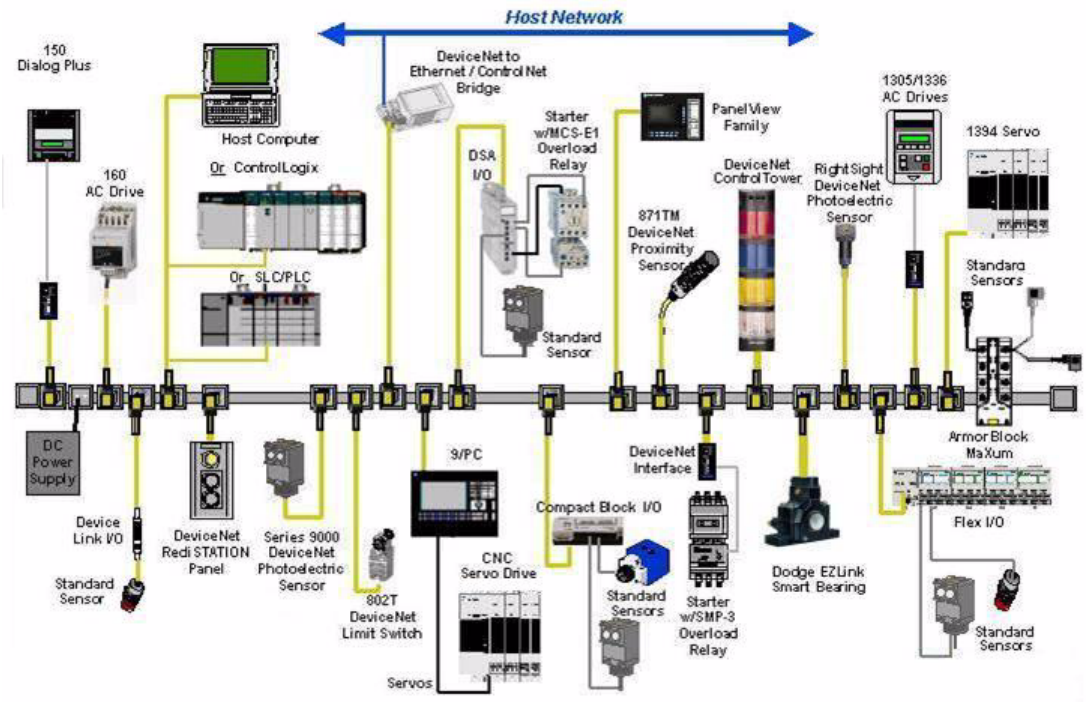


DeviceNet 网络

DeviceNet 网络是面向低级工业设备网络的解决方案，供单设备数据量较低的设备进行实时操作。它最常用在以下类型的配置中：

- 包含多点分布式设备的应用
- 第三方驱动器和其他第三方 “简单” 设备的网络
- 所含设备需要直接连接到网络，并且数据和电源都在同一连接中的系统
- 需要高级诊断信息的环境

典型的 DeviceNet 示例



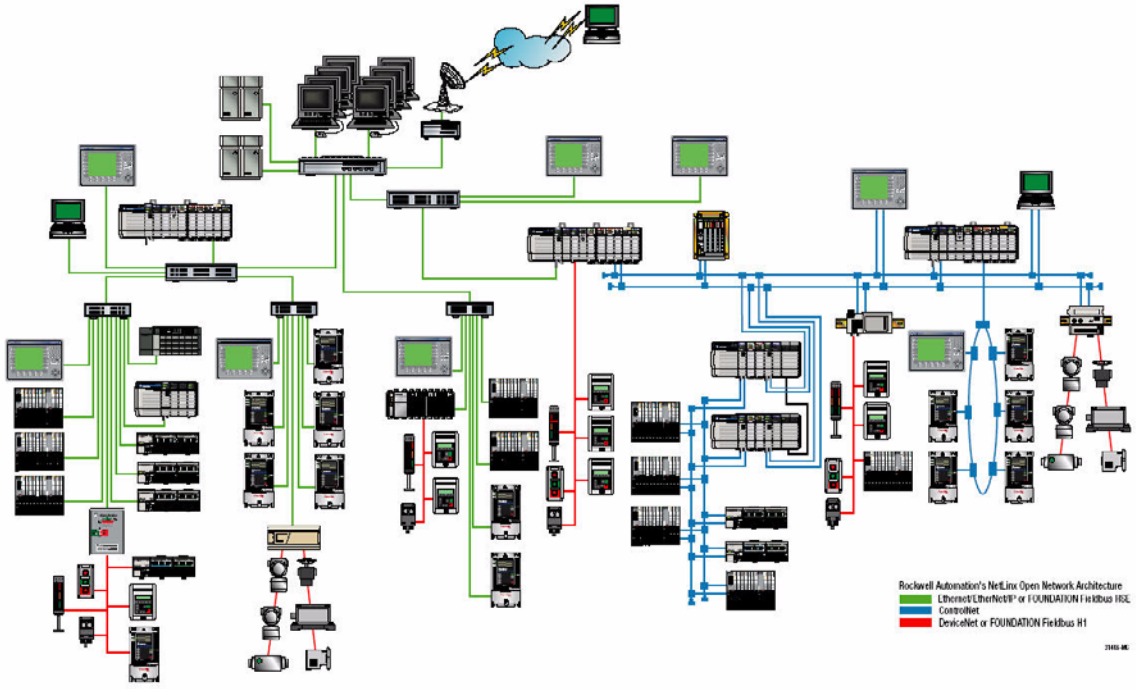
互连 NetLinx 网络

要互连 NetLinx 网络，有两种常用方法。

- 通讯背板，可以一次连接多条网络链路。
- 通讯链接设备，以无缝的方式将两个网络链接在一起。

这两种方式都不需要任何控制器，也不需要进行任何编程工作。

基于 NetLinx 网络的控制系统示例



HMI 的转换

请参考附录 B。

包含分布式控制器的系统的转换

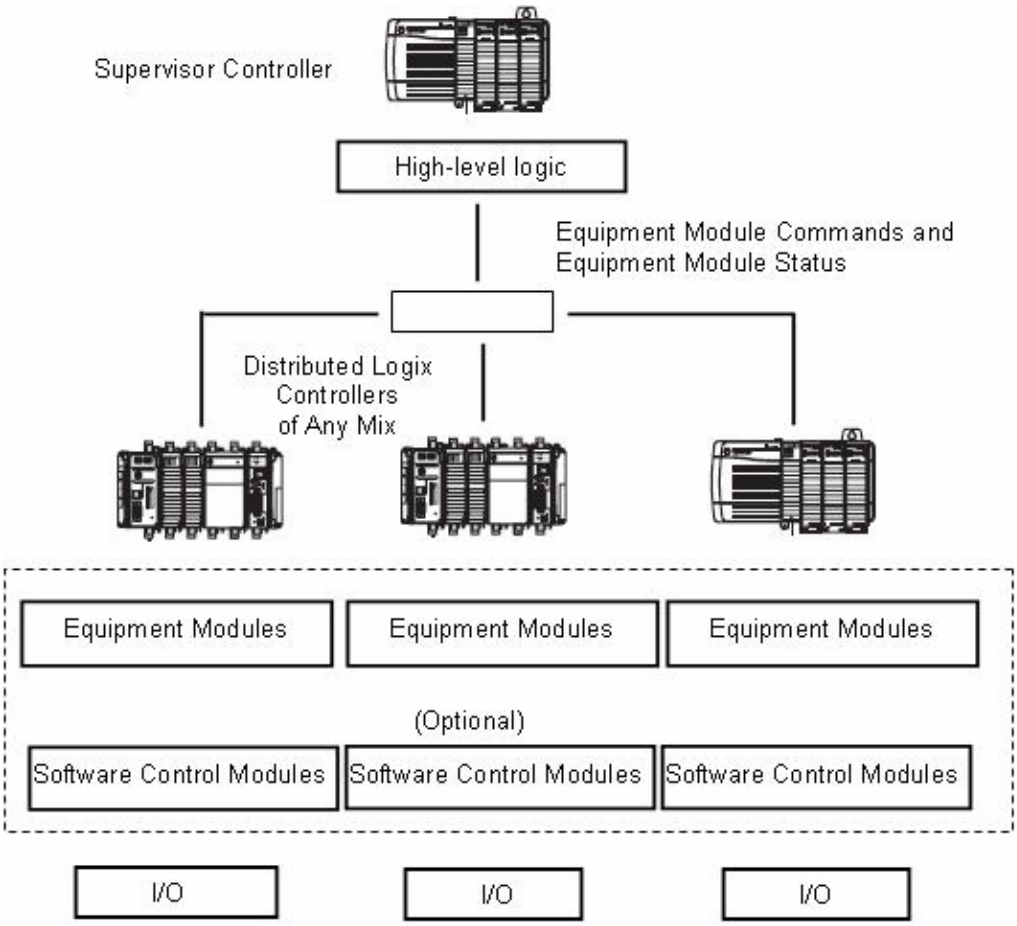
本节介绍：

- 如何使用多个控制器构建包含一组功能单元的常规离散控制应用。
- 如何将类似方法应用于根据 S88 标准设计的过程控制应用。

硬件和软件实施

常规离散控制

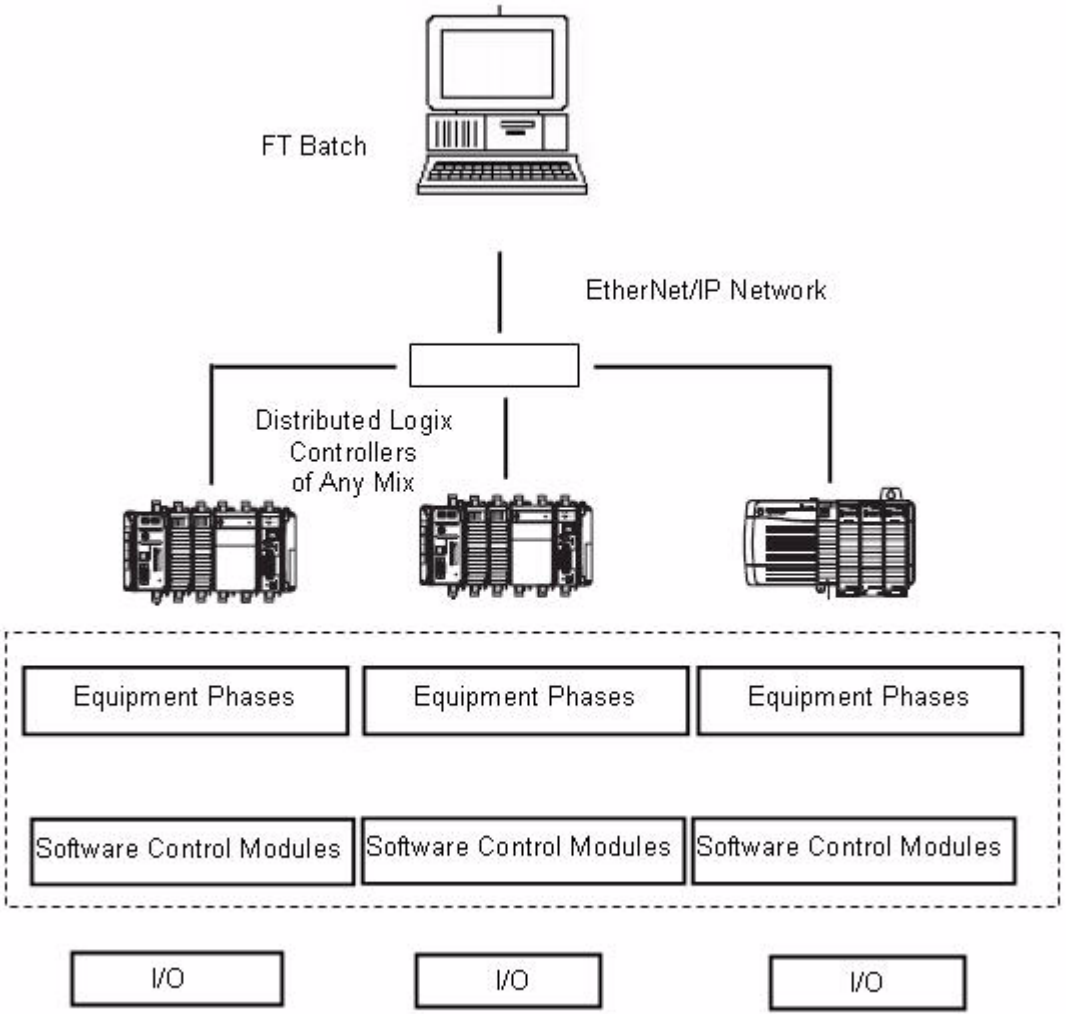
常规离散控制的分布式逻辑的硬件和软件模型如下所示。在这个例子中，监控角色由控制器担任。EtherNet/IP 或 ControlNet 网络可用于互连控制器。生产 - 消费或明示消息传递可用于在系统内交换数据。


Supervisor Controller
High-level logic
Equipment Module Commands and
Equipment Module Status
Distributed Logix
Controllers
of Any Mix
Equipment Modules
Equipment Modules
Equipment Modules
(Optional)
Software Control Modules
Software Control Modules
Software Control Modules
I/O
I/O
I/O

过程控制

下图演示 S88 过程控制应用的硬件和软件结构。PC 将运行 FactoryTalk Batch 软件，这是一种通过配方运行生产批处理的软件包。FactoryTalk Batch 软件位于 PC 中，通过 EtherNet/IP 网络与每个控制器进行通讯。

设备相位是在 PhaseManager 下配置的，如下面的第 2 章所述。设备相位负责执行相位逻辑，通过控制模块与控制系统 I/O 进行通讯。



连接 Siemens 和罗克韦尔自动化设备

在某些情况下，需要将 Siemens 和罗克韦尔自动化设备连接起来。建议使用已加入 Encompass 计划的合作公司的产品。

控制器

Logix 控制器可使用以下设备连接到 S7 网络：

- 机架内置模块。
- 独立通讯网关。

分布式设备

某些罗克韦尔自动化的 I/O 系统、PowerFlex 驱动器和 HMI 终端通过通讯适配器、内置接口或接口模块连接到 Profibus。

注：

S7 用户可能不熟悉的 Logix 功能

简介

本章介绍 S7 用户可能不熟悉的 Logix 功能。

Logix 系统的某些功能比 S7 更容易使用和维护 – 例如，当数据被组织到标签数据库中时，没有绝对地址；而在 S7 中数据项具有绝对地址，绝对地址是程序员在定义的内存区域中选择的。

在其他方面，Logix 的结构与 S7 非常相似，但表示形式不同 – 例如，任务结构实质上是类似于 S7 的组织块。

本章对不相同的功能 （例如标签）进行对比，并对实际相似的功能（例如任务）进行比较。

本章目标：

- 为转换到 Logix 的 S7 用户提供相关信息，以帮助他们更轻松、更快捷地完成设计过程。
- 介绍 Logix 提供了哪些功能可使工程师无需重建控制器固件中已有的内容。

S7 组织块与 Logix 任务的比较

组织块和任务的比较将向 S7 用户介绍 Logix 程序的结构。

组织块和任务非常相似，因为它们都由控制器的操作系统调用，而不是由用户程序调用。在 STEP 7 （和 Logix 中），有三种组织块（Logix 中的任务）。

- 程序循环 OB （Logix 中的组织连续任务），这种 OB 在完成后会重新开始执行。
- 周期性中断 OB （Logix 中的周期性任务），这种 OB 以预先配置的时间周期执行。
- 硬件中断 OB （Logix 中的事件任务）在响应某些硬件激励时执行。

很多 STEP 7 程序员不使用周期性中断 OB。

Logix 提供用户可配置的多任务操作系统，该操作系统可以根据应用的需要分配 CPU。

S7 中的组织块

OB 的类型是由其编号定义的 一 它们可以连续执行 （仅 OB1）、周期性执行 (OB30 一 OB38)、在发生事件时执行 (OB40 一 OB47)，也可在出现某些故障时执行。在 Logix 中，任务没有编号，而是由用户定义的名称来标识。

如果需要，可以为 STEP 7 OB 附加有意义的名称。

OB1 程序循环

OB1 连续循环。当其完成执行时，输出映像表值将发送到输出，输入映像表将根据输出进行更新，然后 OB1 将重新开始。

STEP 7 程序可以不包含 OB1，但如果包含 OB1，则 OB1 将是连续的。

典型 OB1 碎片：

```
Network 3 : Title:
callup valve and motor control module

          CALL  "ValveMotor_Calls"

Network 4 : Title:
callup switch control module

          CALL  "Switch_Calls"

Network 5 : Title:
callup flow totalisers control module

          CALL  "Totaliser_calls"

Network 6 : Title:
callup analogue input control module

          CALL  "AnalogueIn_calls"
```

OB1 是所有连续执行代码的调用层次结构的根。

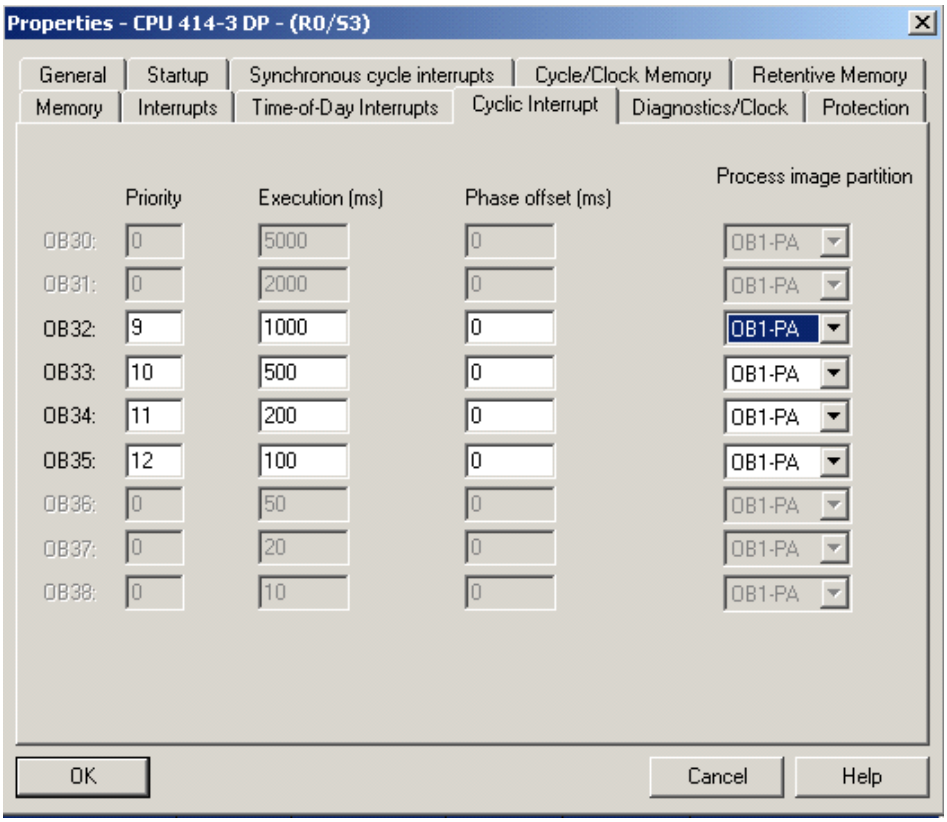
OB1 类似于 （当然只有一种可能）Logix 中的**连续任务**。在 S7 术语中，OB1 称为 “程序循环”。

对于相比 STEP 一 而言更熟悉 Logix 的读者来说，如果了解在 STEP 7 梯形图逻辑中，网络就是 Logix 梯级，这会很有用。在 STEP 7 语句列表中仍然使用网络，但它们只用于改善代码的可读性。它们将代码分成多节，以便添加注释。如果需要，可将所有代码放在一个网络中 – 这样可以很好地编译和运行。

OB30 – OB38 周期性中断

这些 OB 以固定时间间隔 （可配置）执行。它们的优先级也可配置。较高优先级的 OB 将中断所有正在运行的较低优先级的 OB。

如何配置周期性调用的 OB



可用的周期性 OB 的编号取决于控制器的类型。优先级号越低，表示中断优先级越高 （只有 S7 400 控制器允许选择优先级）。Execution (ms) 是 OB 的执行时间长度。相位偏置允许对相互关联的周期性中断触发进行定相。通过过程映像分区选择，可以对 I/O 映像表进行分区，分区只在中断发生时更新 （只有 S7 400 控制器才提供功能）。缺省状况为全表。在 Logix 中，请参见任务 I/O 更新选择和 IOT 命令。

周期性中断 OB 的内容通常与 OB1 的内容相似。它由要在 OB 的周期执行的功能调用和功能块组成。

这些 OB 类似于 Logix 中的**周期性任务**。在 S7 术语中，OB30 — OB38 称为 “周期性中断 OB”。

OB40 — OB47 硬件中断 OB

这些 OB 可配置为在发生输入事件时触发。它们的优先级也可配置。

这些是 Logix 中的**事件任务**。在 S7 术语中，OB40 — OB47 称为 “硬件中断”。

例如，硬件中断 OB （或事件任务）可处理的最简单的硬件事件是数字输入状态的更改。硬件中断 （或事件任务）将保证极快地响应这一更改。

事件任务比硬件中断 OB 更加灵活，它不仅可以从 I/O 触发，还可以通过网络事件、编程指令和运动事件触发。

STEP 7 中的程序结构

典型的程序包含组织块 (OB)、功能块 (FB)、功能 (FC) 和数据块 (DB)。通常会提供系统功能块 (SFB) 和系统功能 (SFC)。

- 功能块和功能是从组织块 （程序循环和 / 或周期性中断）调用的。
- 功能块包含代码，并且与包含 FB 所需的静态数据的数据块关联。除了静态数据外，FB 还有临时数据。在执行过程中，如果逻辑必须保存数值，则会使用 FB。
- 功能包含代码，但不包含静态数据。它有临时数据。当逻辑在单次执行中完成时，会使用 FC 一 不需要保存数值。
- 数据块是存储静态数据的区域。下一节将介绍数据块。
- SFB 和 SFC 是系统功能块和系统功能。可以从 STEP 7 安装随附的库中将它们复制出来，放在项目中。
- 完成这一步骤后，可以从程序的任何位置调用它们。

在 STEP 7 中，没有与 Logix 的程序 / 例程对应的结构。OB 将作为 FB 和 FC 调用链的根，但如何实现则取决于程序员。

Logix 中的任务

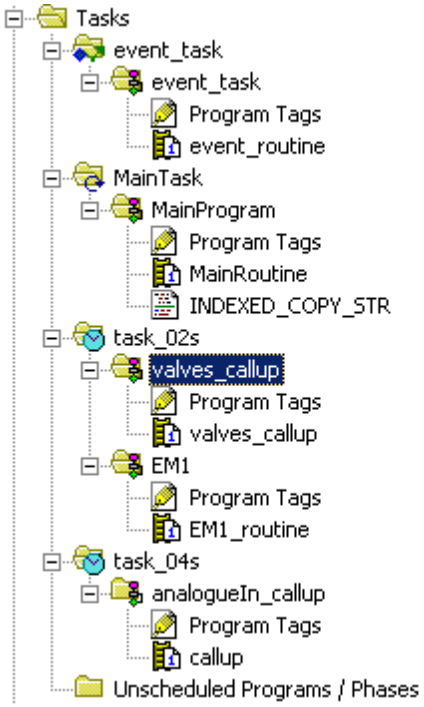
任务是由操作系统调用的。任务为一个或多个程序提供调度安排和优先级。每个程序都包含一个数据段和一个或多个代码例程。

这些任务可以是周期性任务、事件任务或连续任务。可以为每个任务指定优先级。连续任务 （如果有）的优先级总是最低。

Logix 项目有一个默认名称为 MainTask 的任务。此任务可以是连续任务、周期性任务或事件任务。如果需要，可以更改其名称。

Logix 中的任务和程序结构

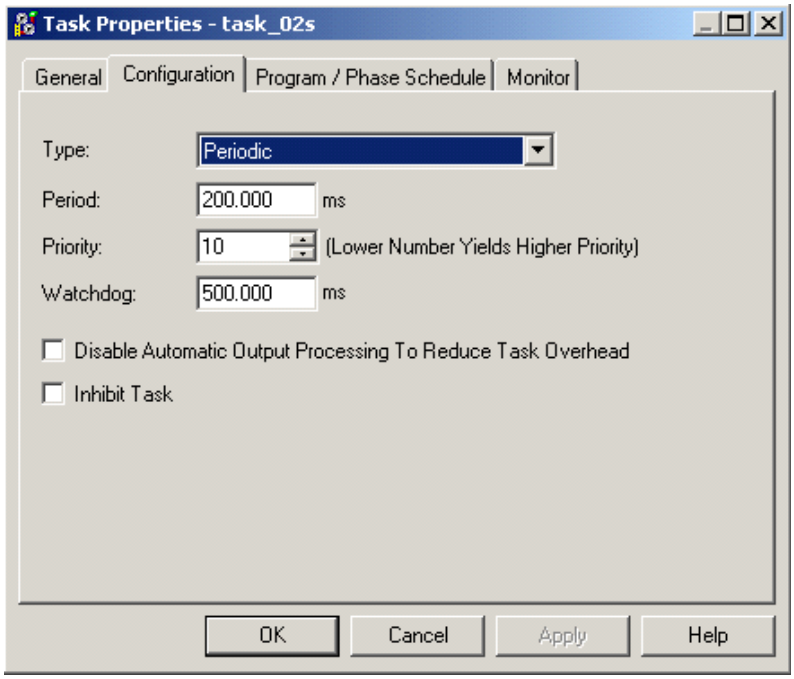
下面的屏幕截图是一个示例 RSLogix 5000 项目树，它有助于说明任务和程序的结构。



在上面的屏幕截图中，“event_task”左侧的图标表示事件任务。“MainTask”左侧的图标表示连续任务，“task_02s”左侧的图标表示周期性任务。

周期性任务

周期性任务将以配置的固定时间间隔触发。周期和优先级的配置如下所示。



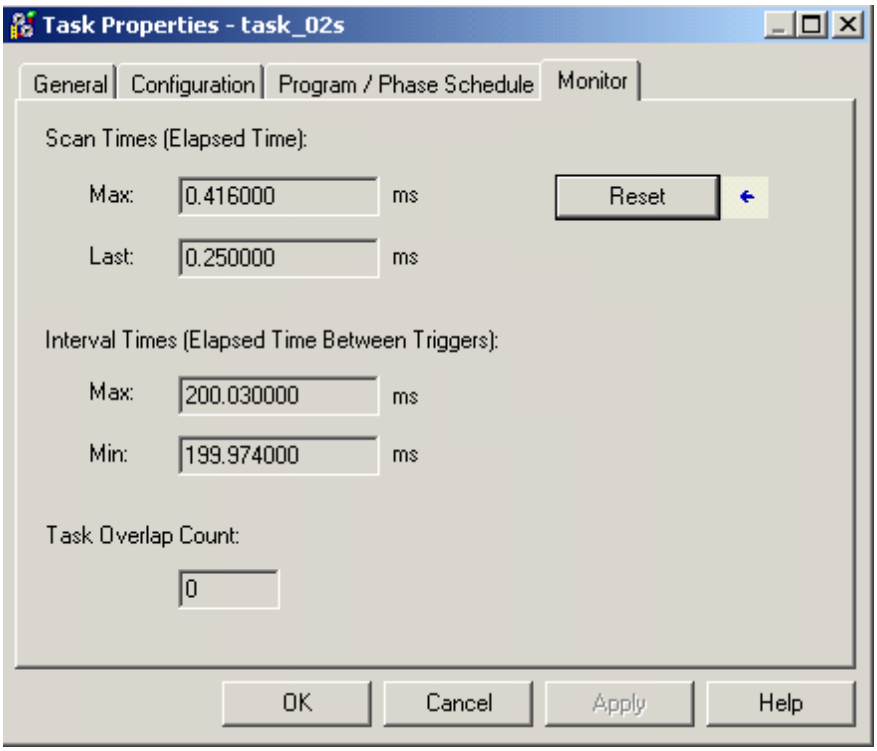
配置类似于在 “OB30 一 OB38 周期性中断”一节中所述的 OB30 一 OB38 配置页。

周期性任务的调度

任务系统的目标：

- 使程序员选择适当的程序执行频率。通过以目标所需频率执行代码，根据应用程序优先级更有效地使用控制器 CPU。
- 使用优先级系统来允许关键任务中断低优先级任务，从而更好地以所需频率执行关键任务。

从 Task Properties /Monitor 中可以方便地检查这些时间。



如果触发器在任务运行时触发，会发生什么情况？

- 如果新触发器所针对的任务比正在运行的任务的优先级高，那么正在运行的任务将被新任务中断，并在高优先级任务完成后继续执行。
- 如果新触发器所针对的任务比正在运行的任务的优先级低，那么正在运行的任务将继续运行，而新任务将等待，直至没有更高优先级的任务运行为止。
- 如果新触发器所针对的任务与正在运行的任务优先级相同，那么控制器将同时运行这两个任务，运行时以 1 毫秒为时间间隔在两者之间切换。
- 如果新触发器所针对的任务就是正在运行的任务，那么新触发器将被丢弃。这就是**重叠**情况。

在任务属性窗口中，会显示自计数器上次复位以来发生重叠的次数。非零值表示中断周期需要延长。

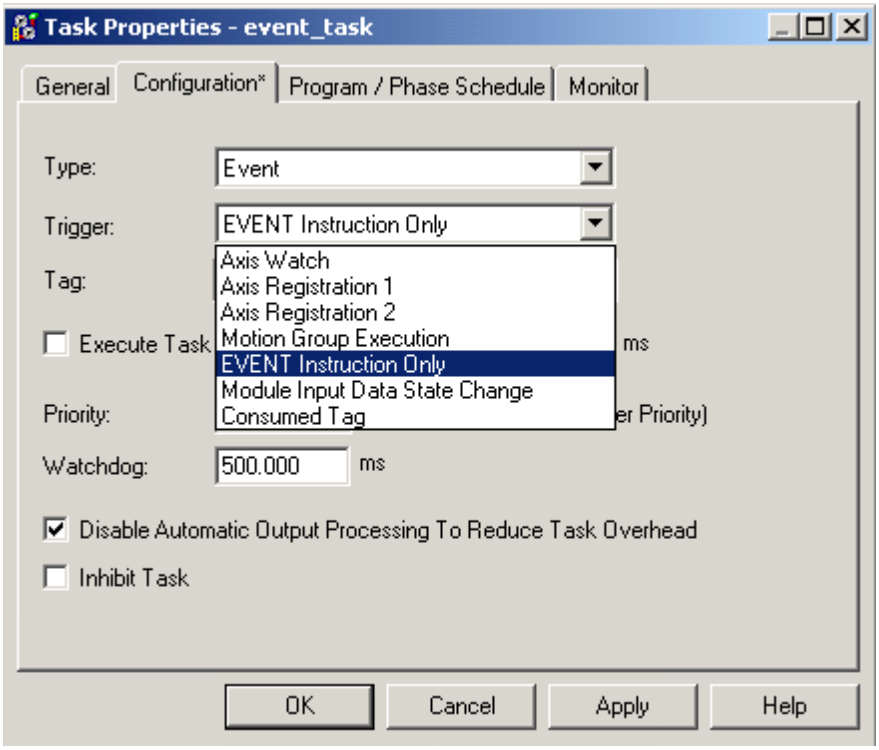
提示 **应避免不必要地切换任务，因为进行不必要的切换会浪费大量处理能力。**

在 Logix 中进行周期性中断编程时，请注意与 STEP 7 之间的以下相似点和不同点：

- 在 STEP 7 中，将从配置为以所选频率执行的 OB 中进行对您希望以该频率执行的功能和功能块的调用。在 Logix 中，将在项目树中将程序和例程插入到 Task 之下。
- 在 STEP 7 和 Logix 中，实际应用程序代码与连续执行任务中的代码不会有很大差异。请注意，通过周期性任务的已知固定频率，程序员可以将简单可变增量转变为计时器。
- 在两种系统中开发和测试代码时，都需要检查是否有重叠。OB 或任务的执行时间必须远远低于其执行周期。
- 检查 Logix 任务的执行时间比较简单。可使用上面所示的任务属性屏幕。在 STEP 7 中，需要获取 OB 开始和结束时的系统时钟，将二值相减，然后将结果存储在变量中以便进行监视。
- 在 S7 控制器中，重叠会使控制器停止，除非添加捕获该故障的故障 OB。Logix 没有这么严格，它只统计重叠次数。
- 在 STEP 7 中，可以分段执行相互关联的周期性 OB。Logix 任务不提供这种功能。

事件任务

事件任务将在配置的触发器事件发生时执行。通常，它们将获得高于周期性任务的优先级。



通过打开任务的 Task Properties 页并选择 Type Event，即可配置事件任务。不同类型的事件任务触发器用于不同的 Logix 控制器。

连续任务

一个 Logix 控制器支持一个连续任务，但是项目可以不包含连续任务。如果愿意，可以在周期性任务和事件任务下运行整个程序。

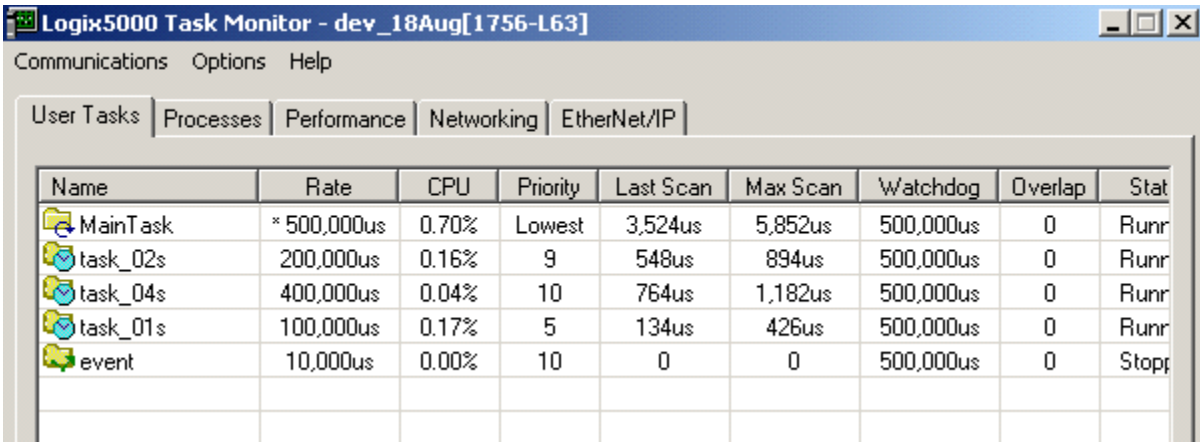
您可以配置连续任务是否在执行结束时更新输出。

如果愿意，可以将用于未调度通讯的 CPU 时间比例调整为专用于连续任务的时间比例。

任务监视器

RSLogix 5000 软件提供一个工具 （称为 Task Monitor），该工具具有分析预定任务等功能。

下面的屏幕截图显示如何在一个表中查看有关控制器任务的信息。



其他选项卡提供有关控制器性能的大量其他系统级信息。该工具作为标配包含在 RSLogix 5000 安装盘中。

使用标签而非地址

在开始使用 Logix 时，S7 用户首先会注意到的主要差别之一是数据没有地址。数据项是在标签数据库中创建的，而 RSLogix 5000 软件在“后台”分配地址。因此，用户不需要理解和管理内存地址。本节介绍两个系统中的数据分配。

S7 中的数据区

S7 控制器中的数据区

地址区	S7 表示法	单位大小
过程映像输入表	I	输入位
	IB	输入字节
	IW	输入字
	ID	输入双字
过程映像输出表	Q	输出位
	QB	输出字节
	QW	输出字
	QD	输出双字
位内存	M	内存位
	MB	内存字节
	MW	内存字
	MD	内存双字

计时器	T	
计数器	C	
数据块	DBX	数据位
	DBB	数据字节
	DBW	数据字

下面各节详细介绍编程中最常用的两个区 – 位内存和数据块。

位内存

“位内存”位置表示为 M*x*，例如：

- **M5.3** 是位。
- **MB6** 是字节 (BYTE)。
- **MW8** 是 16 位字 (WORD)。
- **MW10** 是 32 位字 (DWORD)。

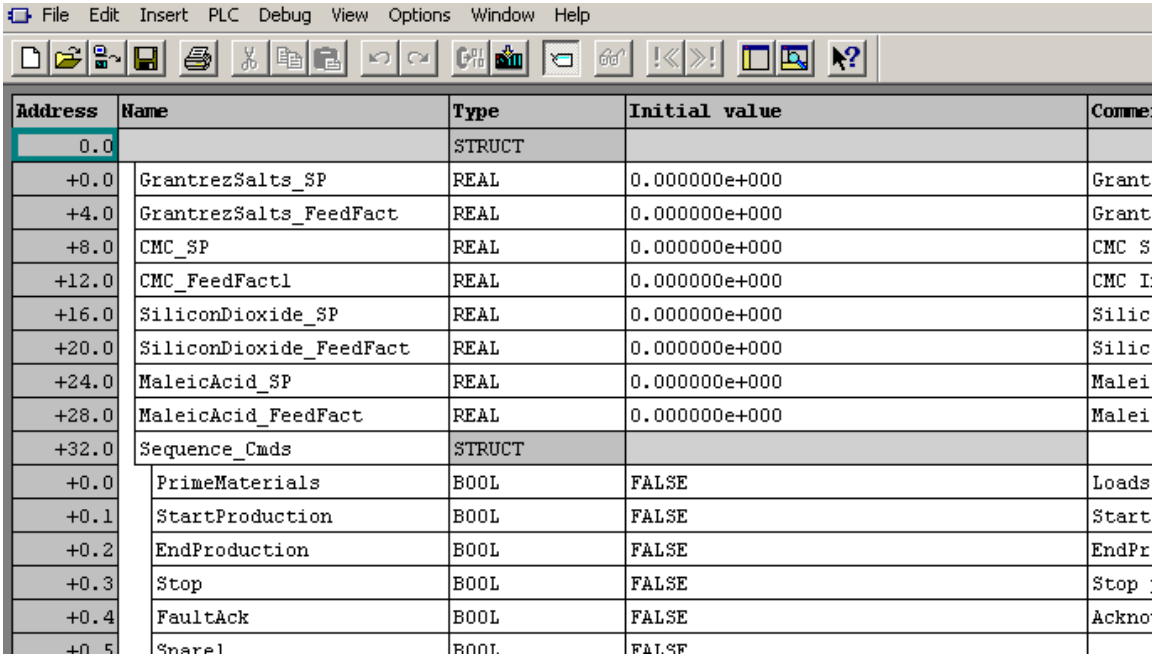
位内存位置可在符号表 （类似于 PLC-5 或 SLC 符号表）中标记，如下面的屏幕截图所示。

Status	Symbol	Address	Data type	Comment
	EXT_ZONE2_ON	Q 28.2	BOOL	EXTRUDER ZONE2 ON
	EXT_ZONE3_ON	Q 28.3	BOOL	EXTRUDER ZONE3 ON
	EXT_ZONE4_ON	Q 28.4	BOOL	EXTRUDER ZONE4 ON
	EXT_ZONE5_ON	Q 28.5	BOOL	EXTRUDER ZONE5 ON
	FALSE	M 0.0	BOOL	
	FlowTotaliser	UDT 10	UDT 10	
	FSL24001	I 10.0	BOOL	
	FSL24002	I 41.0	BOOL	
	FSL24003	I 10.6	BOOL	
	FT24001	PIW 530	INT	
	FT24002	PIW 526	INT	
	FT24006	PIW 570	INT	
	GET_INDEXED_REFE...	FC 111	FC 111	
	Global_Data	DB 99	DB 99	
	IFIX_alarms	DB 71	DB 71	
	INDEXED_COMPARE	FC 102	FC 102	
	INDEXED_COPY	FC 101	FC 101	
	Interlocks_Handler	FB 70	FB 70	
	Interrupt_Execution	OB 35	OB 35	
	JUNK_BIT	M 0.2	BOOL	
	KTRON_CmdsFromM...	DB 33	DB 33	Commands & Setpoints From Manufacturing PLC
	KTRON_StatusToMan...	DB 32	DB 32	Feeder Status to ManufacturingPLC
	LF24001A	Q 17.0	BOOL	PIG LAUNCH VALVE
	LF24002A	Q 18.5	BOOL	PIG LAUNCHER
	LF24003A	Q 10.4	BOOL	PIG LAINCHER

数据块

数据块的状态与其他块 （组织块、功能块和功能）相似，区别之处在于它们包含数据而不是程序代码。数据块中的内存是静态的 – 数据保持其值，直到更改为止。

数据块示例



Address	Name	Type	Initial value	Comme
0.0		STRUCT		
+0.0	GrantrezSalts_SP	REAL	0.000000e+000	Grant
+4.0	GrantrezSalts_FeedFact	REAL	0.000000e+000	Grant
+8.0	CMC_SP	REAL	0.000000e+000	CMC S
+12.0	CMC_FeedFactl	REAL	0.000000e+000	CMC I
+16.0	SiliconDioxide_SP	REAL	0.000000e+000	Silic
+20.0	SiliconDioxide_FeedFact	REAL	0.000000e+000	Silic
+24.0	MaleicAcid_SP	REAL	0.000000e+000	Malei
+28.0	MaleicAcid_FeedFact	REAL	0.000000e+000	Malei
+32.0	Sequence_Cmds	STRUCT		
+0.0	PrimeMaterials	BOOL	FALSE	Loads
+0.1	StartProduction	BOOL	FALSE	Start
+0.2	EndProduction	BOOL	FALSE	EndPr
+0.3	Stop	BOOL	FALSE	Stop
+0.4	FaultAck	BOOL	FALSE	Ackno
+0.5	Spare1	BOOL	FALSE	

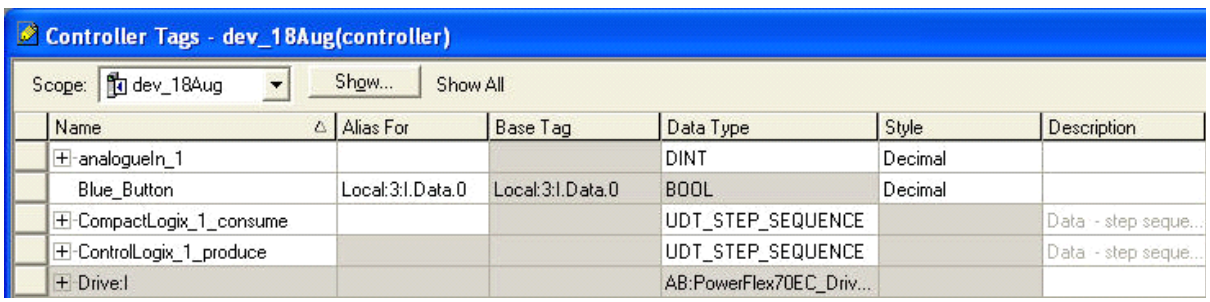
符号表中不显示数据块符号，但显示数据块的名称。

数据块可分配来保存功能块所用的数据。这种数据块称为实例数据块。

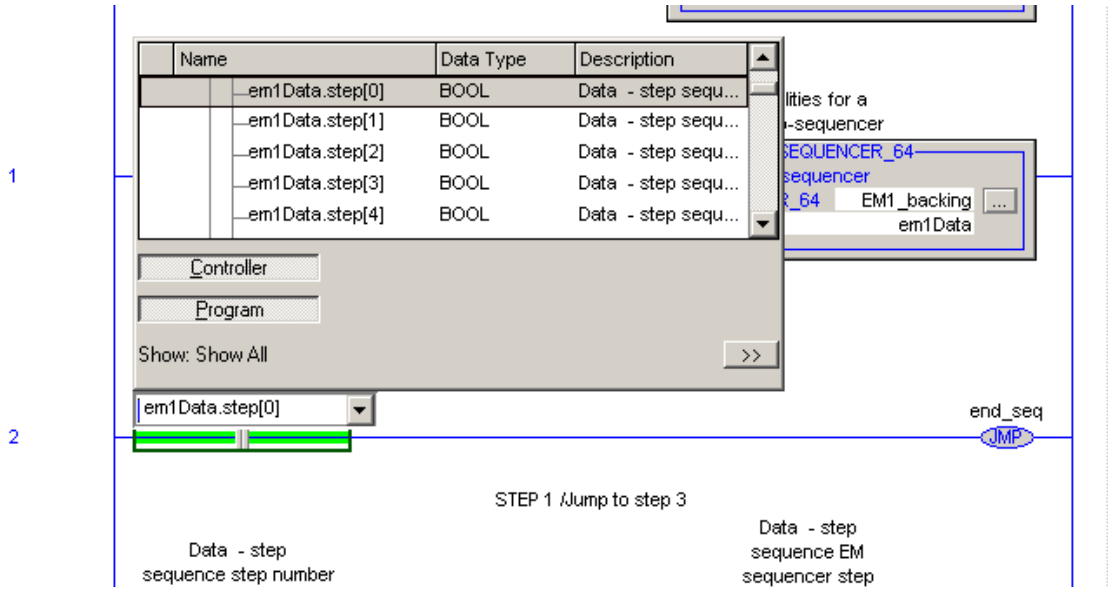
Logix 中的数据

在 RSLogix 5000 编程环境中，数据设置在标签数据库中。对于程序员而言，内存地址是隐藏的，这使程序员的工作更为轻松。

标签数据库



在编程时，从下拉菜单中选择一个标签



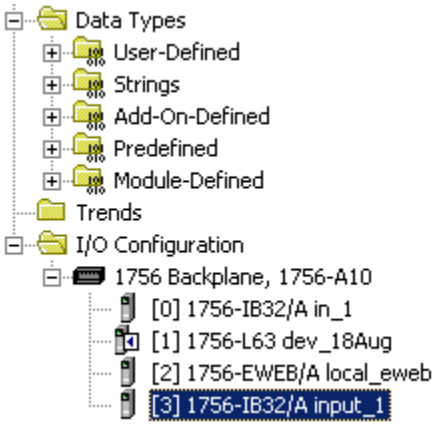
在 Logix 中，每个程序都与一个**控制器作用域标签数据库**和多个**程序作用域标签数据库**相关联。

- 控制器作用域数据库中的标签是全局的，程序任何部分中的例程都可以访问。
- 程序作用域标签只能由该程序中的例程访问。

I/O 和别名标签

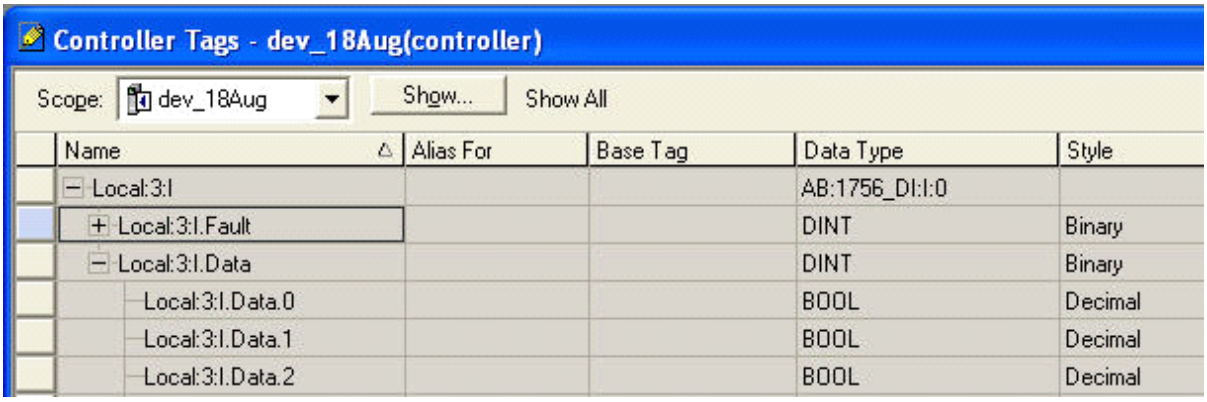
别名标签用于表示另一个标签，而这两个标签的数值是相同的。别名的目的之一是引用下述 I/O 标签。

通过将模块添加到项目文件夹中的控制器背板，可将 I/O 模块添加到项目。

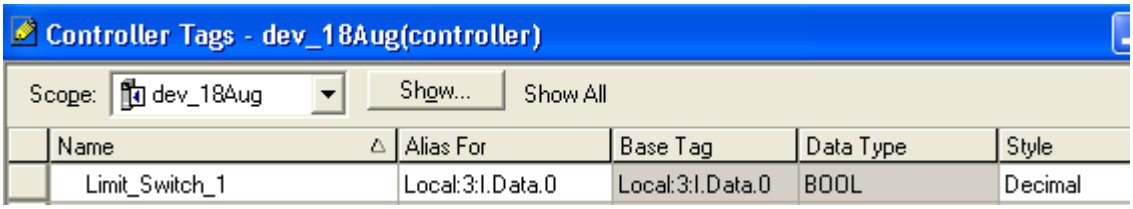


本例中，一个 32 点输入卡已添加到插槽 3。插槽号在行首的方括号中。“1756-IB32/A”是卡的部件号。“input_1”是卡的名称，它是在卡首次添加到机架时配置的。

添加卡之后，RSLogix 5000 软件将自动生成相关的设备配置文件标签，并输入到控制器作用域标签数据库。它们是下面所示的 Local:3:I 输入和 Local:3:C 配置标签。



可以用描述性更强的名称来创建新别名标签。例如，可以为第一个输入创建别名 Limit_Switch_1，该名称对该输入进行了具体描述。



在 STEP 7 中，硬件配置工具将在 I/O 卡添加到系统时为其分配地址。例如，数字输入卡可能分配字节 I16 和 I17。然后，程序员将确定每个输入的位地址，在符号表中为其输入对应的名称。执行这些操作后，程序将自动建立关联 I16.5 = “ZSC2036”。

编程语言

本节介绍 STEP 7 和 RSLogix 5000 软件中使用的编程语言。所有语言都不是标配的；它们取决于所购软件的版本。选择最适合于任务的 Logix 语言可使程序设计更轻松、代码编写更快速，程序也更易于理解。

S7 和 Logix 语言之间有一个重要的区别。在 S7 中，语句列表是控制器的“本机”语言。其他语言会转换为 STL。在 Logix 中，所有语言都是控制器中的“本机”语言？ 每一种都无需引用任何其他语言即可编译。优点在于从控制器上载程序时，可以查看原始编写语言形式的程序。

STEP 7 有三种标准语言：

- 语句列表 (STL) – 可描述为高级汇编程序。
- 梯形图逻辑 (LAD)
- 功能块图 (FBD)

某些可选语言：

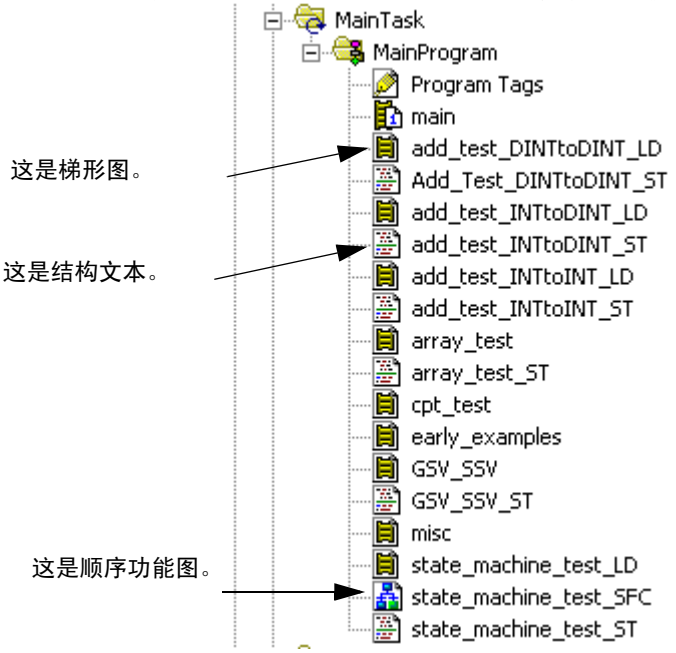
- 结构文本 (ST)
- CFC – 过程类型应用的连续流程图
- HiGraph – 通过图形软件的顺序控制
- ML （运动语言）– 类似于早期罗克韦尔自动化专用 1394 运动控制器的 GML

程序可以由用不同语言编写的功能块和功能所组成。

RSLogix 5000 软件有四种编程语言：

- 梯形图 (LD) – 相当于 Siemens LD，附带扩展指令集。
- 结构文本 (ST) – 相当于 Siemens ST
- 功能块图 (FBD) – 相当于 Siemens CFC
- 顺序功能图 (SFC) – 相当于 Siemens hiGraph。

例程 （Logix 中的基本代码段）可以采用上述任何一种语言，程序可以由用不同语言编写的例程构成。下面的屏幕截图提供了示例。



Logix 梯形图

传统上，梯形图用于实现布尔组合逻辑。在 Logix 中，它还可用于顺序逻辑、运动、数据操作和数学计算 （尽管这些任务使用其他语言可能更为方便）。

Logix 结构文本

结构文本是高级过程语言，凡是具有 Basic、Pascal 或任何一种“C”语言使用经验的人，都可以轻松学习该语言。它主要用于数据操作和数学计算 （尽管使用 ST 可以方便地进行运动逻辑、组合逻辑和顺序逻辑的编程）。

Logix 功能块图

功能块图以图形形式描述关联输入变量和输出变量的功能 （布尔或算术）。输入和输出变量通过连接线连接到块。一个块的输出也可连接到另一个块的输入。

建议在 FBD 中编写 PID 循环。对于过程控制，这是最方便的语言。

Logix 顺序功能图

SFC 是一个图形化工具，用于将顺序逻辑描述为一组状态和转换。可以为输出分配一种状态和多个布尔条件，这些条件确定是否转换为已定义的其他状态。

STEP 7 代码到 Logix 的转换

- 如果要将 STEP 7 梯形图逻辑代码转换为 Logix，应首选 LD。LD 的意义在两种系统中很相似。
- 如果要将 STEP 7 功能块图代码转换为 Logix，应首选 FBD。
- 注意，标准 Logix FBD 比 STEP 7 FBD 更高级，相当于可选的 STEP 7 语言 CFC。
- 如果要将 STEP 7 语句列表代码转换为 Logix，则最合适的语言将取决于 STL 块的特性。如果 STL 块主要包含布尔求值，LD

则可能是最佳的转换 Logix 语言。如果 STL 块包含指针来存取和操作数据，或者执行数学计算，ST 则可能是最佳的转换 Logix 语言。如果 STL 块包含顺序逻辑，则应考虑 SFC（尽管用 ST 和 LD 也可以方便地实现顺序逻辑）。

使用数组而非指针

在 STEP 7 中，可以像在 Pascal 或 C 中一样定义数组，但基本语言（STL、LD 和 FBD）未提供用于存取数组的高级支持。相反，必须构造指针例程。

STEP 7 库功能缺少对数组存取的支持。习惯使用指针的程序员可以编写自己的功能，如 FC101 “INDEXED_COPY”（请参见下文），但这需要技巧和时间。

STEP 7 中的 “INDEXED_COPY” 与 Logix 指令 COP 的功能相同，都用于带索引复制。

```
CALL  "INDEXED_COPY"                FC101
 indexSrc:=#index_in
 source  :="Instance_FB2".table     P#DB4.DBX0.0
 indexDst:=1
 dest    :="Instance_FB2".target    P#DB4.DBX96.0
 len     :=8
```

下面的 FC111 将存取数组。

```
CALL  "GET_INDEXED_REFERENCE"       FC111
 refArray  :="Instance_FB2".table   P#DB4.DBX0.0
 index     :=#index_in
 byteIncr  :=32
 startIndex:=TRUE
 retVal    :=#ptr
```

指向对象的指针是在参数 #ptr 中返回的，对该参数解除引用可以获得数据。

在 Logix 中，可以用高级计算机语言的常用方法来定义和存取数组，如下面的代码段所示。

```
// copy a string from a table of strings #table
// to a target string #target. The index is #index_in

COP(table[index_in], target, target.LEN);
```

附加指令

附加指令摘要

附加指令相当于 STEP 7 功能块，它有专用数据和高级参数选项。尤其是 INOUT 参数类型或 “按引用传递”可以有效地将数据结构传递给代码。

由于附加指令与 STEP 7 功能块非常相似，因此要转换到 Logix 的 S7 程序员可以非常方便地随时使用附加指令。

FB 和附加指令之间的比较：

- 两者都可以作为命名功能从程序中的任何地方进行调用。
- 两者都包含静态数据的专用数据区，尽管在 STEP 7 中并不是真正的专用。
- STEP 7 功能块也有临时数据区。
- 在附加指令中，本地静态数据具有同样的功能。

两者都有三种参数 ？ 输入 （按值传递）、输出 （按值传递）和输入 - 输出 （按引用传递）。按引用传递参数有很大的优点，因为它可以有效地传递大型数据结构。

通过在变更时记录时间戳和 Windows 用户名，附加指令将自动维护变更历史记录。STEP 7 功能块没有这一功能。

通过附加指令，可以将预扫描例程配置为在控制器从程序模式转为运行模式时运行，或者在运行模式下加电时运行。在这类情况下，预扫描例程将运行一次，通常可用于初始化数据。在 STEP 7 中，组织块 OB100 的功能相同，但预扫描代码不能专门附加于 FB。

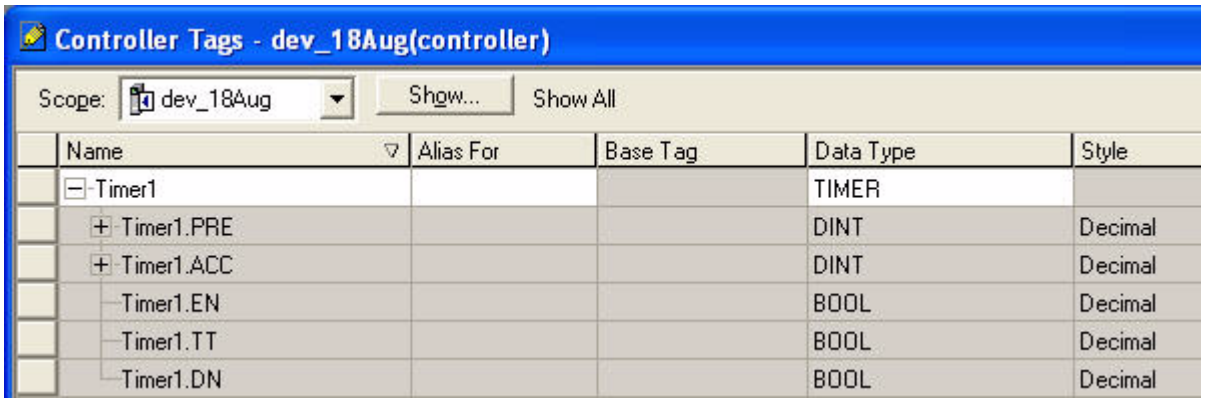
如果从 SFC 步骤中调用附加指令，并且 SFC 配置为 “自动复位”，则附加指令中定义的后扫描例程将在 SFC 退出该步骤时执行一次。它可以用于复位数据。STEP 7 FB 没有内置的等效功能 （尽管通过编程很容易实现）。

附加指令可以有一个 EnableInFalse 例程，当附加指令调用的梯级条件为 false 时，将调用该例程 （如果有）。在这种情况下，输入和输出参数将传递值。STEP 7 FB 没有等效功能。

第 4 章将进一步讨论附加指令。

支持标签

很多指令和数据类型都使用支持标签 ？ 专为指令的实例或者要实例化的数据类型而创建的标签。附加指令、计时器、计数器、消息和 PID 控制都使用支持标签。每当创建某个类型的标签时，RSLogix 5000 软件都会生成对应的元素结构，因此您无需自己创建这些元素。



通用工业协议 (CIP)

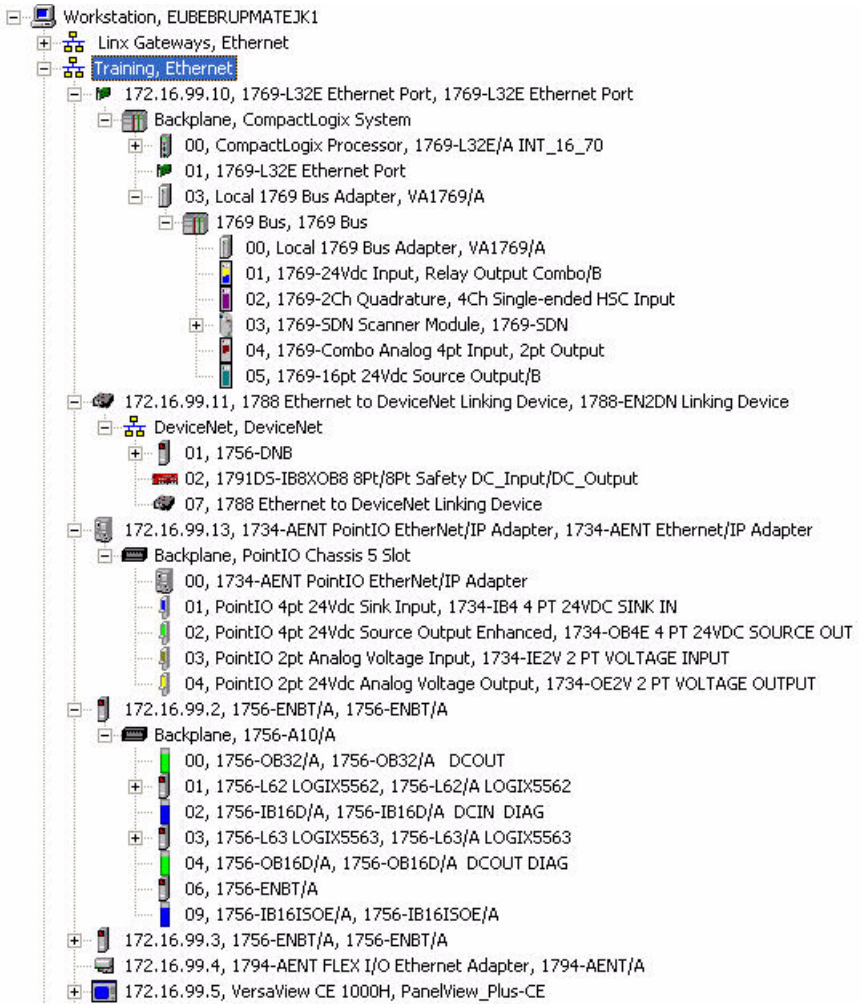
Logix 使用三种主要网络 ？ Ethernet/IP、ControlNet 和 DeviceNet。每一种网络都具有适合于不同应用领域的特性。三种网络类型都使用一种协议，即 “通用工业协议”。

CIP 可以通过 Logix 所支持的这三种网络类型中的任何一种传输数据，并且所有三种网络的配置和编程接口几乎完全相同。此外，即使网络由这三种网络类型中的多种类型组成，数据仍可以在网络中传输，程序员无需转换协议。

在 “传统”S7 中，两种主要的协议是工业以太网 （用于通过网络连接到 IT 和其他控制器）和 Profibus DP （用于通过网络连接到现场系统）。这两种协议分别应用于硬件级和数据级。对于最新的 S7 硬件和软件，“Profinet CBA”将工业以太网、Profinet 和 Profibus 集成在一起。

查看网络

S7 用户可能会对 Logix 网络配置和管理留下深刻印象。作为示例，下面的树显示了实际连接到系统的设备。此树是在线生成的 - 未进行任何配置。



第 1 章将进一步介绍网络。

控制器之间的数据交换

STEP 7 中的发送 / 接收

若要在 STEP 7 中准备控制器对控制器通讯，需要执行以下步骤。

1. 在 STEP 7 组件 NetPro 中以图形方式配置远程工作站。
2. 在 NetPro 中建立连接表，指定每个连接的协议和参数。

3. 将库功能 FC5 AG_SEND 和 FC6 AG_RECV 复制到项目中。

4. 从用户程序中调用 AG_SEND 和 AG_RECV，调用时指定连接参数以及用于发送和接收数据的数据区。

Logix 中的生产 / 消费标签

生产 / 消费标签是一种方式，即每隔定义的时间周期就在联网的 Logix 控制器之间传输关键数据。生产 / 消费标签可通过 Ethernet/IP 或 ControlNet 传输，可以在 ControlLogix 控制器的背板上传输。

生产 / 消费标签是在创建标签时配置为生产或消费的标签。如果标签标记为生产，则它的值将通过多播方式传送到控制器所连接的 EtherNet/IP 或 ControlNet 网络。如果标记为消费，则将为标签提供所需数据的控制器标识为配置的组成部分，并且消费标签将从该控制器的对应生产标签获得其值。

发送和接收各自具有单独的通道。更改消费标签的值不会影响生产标签。这类似于 S7 中的控制器对控制器通讯，但不同于控制器对 SCADA 通讯，对于后者，任何更改都会在另一端反映出来。

设置生产 / 消费连接不需要进行编程。这一点与 S7 不同，在 S7 中，控制器对控制器 (SEND/RECEIVE) 通讯需要编写一些代码。

用户定义的数据类型

在 Logix 中，可以配置用户定义数据类型。这样可将复杂数据类型结构声明为一个类型。然后，即可在程序中定义该类型的实例。

Logix 用户定义数据类型的配置和用法与 STEP 7 用户定义数据类型非常相似。

Logix UDT

Name: UDT_RAMPER

Description: Ramps a real variable from its current valuse to a new value at a specified rate.

Members: Data Type Size: 28 byte(s)

Name	Data Type	Style	Description
initial_output	REAL	Float	saved initial output
increment	REAL	Float	calculated increment
RAMP_RATE_ABS	REAL	Float	per second - (set always +ve)
RAMP_TARGET	REAL	Float	final value - (set)
change	REAL	Float	calculated change over ramp
counter	DINT	Decimal	internal counter
complete	BOOL	Decimal	ramping is complete
_enable	BOOL	Decimal	for enable one shot
enabled	BOOL	Decimal	ramper enabled

异步 I/O 更新

在 Logix 系统中，I/O 根据程序执行周期进行异步更新，这与 S7 中使用的传统 PLC 方法不同；在 S7 中，I/O 映像表在周期开始时更新，输入值在程序执行期间不会更改。

Logix 程序员需要考虑是否需要缓冲输入数据，以使其值在程序执行期间保持不变。

常见做法是只使用一次输入，即将输入作为参数传递给代码模块。程序中的任何其他位置都不会使用这些输入。这样就不必进行缓冲。请参见第 4 章中的控制模块示例。

DINT 数据类型

Logix 控制器操作 DINT （32 位整型）标签比操作 INT （16 位整型）或 SINT （8 位整型）更加高效。即使要处理的数值范围适合于 INT 或 SINT，也应尽可能使用 DINT。提供这些数据类型是出于 IEC61131-3 兼容性的原因，不过在程序使用这些类型之前，会在内部转换为 DINT，因此在大多数情况下，代码将更加高效地执行。

相位管理器

STEP 7 中的相位管理

STEP 7 没有内置的工具来执行相位管理。必须在一组例程（通常称为 PLI 或相位逻辑接口）中编写必要的结构。基于 S88 的 PLI 程序组件为：

- 行为符合 S88 状态模型的步骤定序器。某些步骤或某些一系列的步骤定义 S88 状态。定序器命令也按照 S88 指定，并且定序器只在状态模型允许时响应。具有这些属性的定序器称为相位。
- 每个相位的一组数据，用于记录该相位的状态并接收从配方管理器传入的命令。配方管理器通过这些数据进行通讯。数据的格式取决于配方管理器。
- 逻辑模块，该模块将相位状态转换为配方管理器所要求的格式，并将来自配方管理器的命令转换为相位命令。

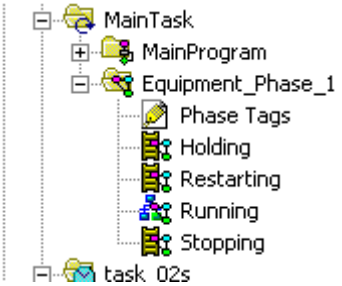
Logix 中的 PhaseManager

在 S88 设备相位中，有指定的相位状态以及状态之间的转换。PhaseManager 是 RSLogix 5000 软件的一个功能，用于执行三种任务：

- 将每个相位状态的代码分配给不同的例程。
- 在 “后台”运行状态机器，由状态机器处理相位状态之间的转换。
- 使用一组 Logix 命令管理相位的运行。

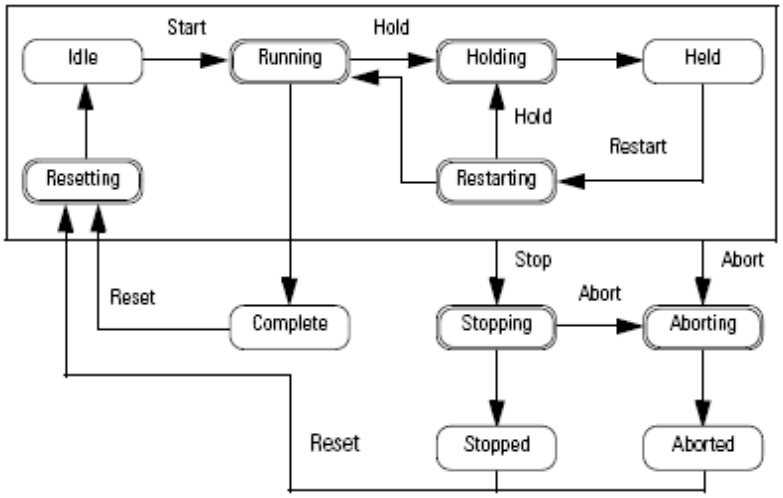
它用在多种应用空间中，包括但不限于过程控制和封装，因为它实现了设备 / 装置控制的明确分离和过程控制的明确分离，因此使代码更加模块化、更加高效，对于更大型的标准化系统更是如此。

项目树中的设备相位



相位的每种状态的代码都可以用任何 Logix 语言编写。

下面是相位状态机器。它与 S88 状态模型几乎完全相同。



如果编写了符合 S88 的 STEP 7 相位管理器 / PLI 例程，希望将它转换为 Logix，那么使用 Logix PhaseManager 就不必进行转换。

协调系统时间 (CST)

S7 有一个用 32 位表示的系统时钟，以毫秒为计数单位。通过向操作系统发出调用可以获得 （和存储）它的值，这对于精确测量时间间隔非常有用。

Logix 使用 64 位 (bit) 数的协调系统时间来测量自控制器上次启动以来的毫秒数。与 S7 一样，通过向操作系统发出调用来获得 CST 值，也可以测量时间间隔。它为多 CPU 系统的时钟同步、准确运动控制功能、切换到 100 微秒精确度的预定输出、输入事件时间戳、预定模拟采样、安全 I/O 监视和通讯、运动镜头位置计算和时钟时间提供了基础。

时间戳输入

时间戳是一种功能，它记录输入数据的更改，同时记录发生更改的相对时间。通过数字输入模块，可以配置数据更改的时间戳。可以使用 CST 时间戳来比较数据样本之间的相对时间。

这样，在将输入信号与应用的时间参考值相关联时 （例如在运动控制中经常用到），程序员可以获得无与伦比的精确性，而且不会对通讯和逻辑处理系统以及相关应用代码造成巨大负担。

预定输出

通过数字输出模块，可以将模块配置为在预定的时间设置输出。

这样，在将输出与应用的时间参考值相关联时 （例如，运动控制中的轴定位或过程控制功能），程序员可以获得无与伦比的精确性，而且不会对通讯和逻辑处理系统以及相关应用代码造成巨大负担。

无临时变量

S7 有一个称为 “临时变量”的变量类别。它们的作用域是定义它们的程序块，其有效期是定义它们的程序块的执行时间。

Logix 没有对应的临时变量。所有变量都是静态的 一 它们一直保持其值，直到更改为止。

若要实现通常针对 S7 应用的功能，举例来说，可以使用下列方法之一：

- 使用程序作用域标签。
- 如果要使用附加指令进行编程，则使用本地标签 （附加指令数据的组成部分）。

不需要累加器或特殊寄存器

如果使用 STEP 7 语句列表编程，就会非常熟悉累加器和 AR1 和 AR2 指针寄存器。在 Logix 中没有对应的功能。所有操作数都是标签。

若要实现通常针对 S7 应用的功能，举例来说，可以使用下列方法之一：

- 使用程序作用域标签。
- 如果要使用附加指令进行编程，可使用本地标签 （附加指令数据的组成部分）。
- 考虑是否需要 S7 累加器和特殊寄存器的对应 Logix 功能。S7 语句列表具有低级的特性，因此使用累加器和特殊寄存器，在结构文本等语言中，不可能需要它们。

系统软件和标准功能的转换

简介

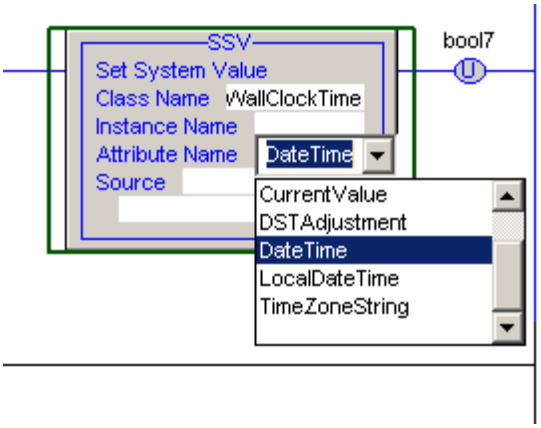
本章列出了更常用的 S7 系统功能，说明如何在 Logix 中实现等效功能，并提供几个具体示例。

本章的目的是使您了解 Logix 中的专用指令，以免浪费时间去开发已有的解决方案。

Logix 系统功能

在 Logix 中，大多数 S7 系统功能的等效项为 GSV （获取系统值）和 SSV （设置系统值）指令。这些指令存取 Logix 控制器内置的对象 （类别、实例和属性）层次结构。如果要进行 GSV 和 SSV 编程，可以通过下拉菜单选择参数。

SSV 指令



学习 GSV 和 SSV 基础知识后，Logix 新用户会发现访问操作系统比使用 S7 SFC 更为方便。

复制

用于复制复杂数据结构 – 用户数据类型的实例数组。

S7	备注	Logix	备注
SFC20 BLKMOV	使用 BLKMOV 时，地址必须在编译时进行定义。	COP 指令	如果 COP 用于在数组之间进行复制，则块 （源或目标）的开头可能包含一个数组索引，以便对在运行时计算值的元素进行寻址。
SFC81 UBLKMOV	不间断版本 ？ 确保在复制过程中不会改变源数据。	CPS （指令）	不间断版本 ？ 确保在复制过程中不会改变源数据。
SFC14 DPRD_DAT	如果 Profibus DP 设备的通讯数据区 > 4 字节，则 SFC 可确保读取的一致性。	CPS（ControlNet 和 Ethernet /IP）	DeviceNet 不需要
SFC15 DPWR_DAT	如果 Profibus DP 设备的通讯数据区 > 4 字节，则 SFC 可确保写入的一致性。	CPS（ControlNet 和 Ethernet /IP）	DeviceNet 不需要

日期与时间的设置和读取

两种系统的控制器都有一个可以进行读取或设置的实时时钟。

S7	备注	Logix	备注
SFC0 SET_CLK	在类型为 DT (DateTime) 的实例中传递的值	SSV （设置系统值）	SSV 类 - WallClockTime SSV 属性 - DateTime SSV 源 - 指定 DINT[7] 的元素 [0]
SFC1 READ_CLK	在类型为 DT (DateTime) 的实例中返回的值	GSV （获取系统值）	GSV 类 - WallClockTime GSV 属性 - DateTime GSV 目标 ? DINT[7] 的元素 [0]

读取系统时间

这两种系统都有系统时钟，系统时钟在控制器启动时启动。在 S7 系统中，时间以毫秒为单位，而在 Logix 中，时间以微秒为单位。

S7	备注	Logix	备注
SFC64 TIME_TCK	返回介于 0 至 2.31 毫秒的系统时间	GSV （获取系统值）	返回介于 0 至 2.63 微秒的系统时间 GSV 类 - CST GSV 属性 - CurrentValue GSV 目标 - 指定 DINT[2] 的元素 [0] DINT[0] - 低 32 位 DINT[1] - 高 32 位

处理中断

用户程序可以通过调用系统功能来启用和禁用中断。

S7	备注	Logix	备注
SFC39 DIS_IRT	禁用由指定的 OB 处理的中断。中断请求丢失。	SSV 禁止指定的任务。	SSV 类 - Task SSV 实例 - Task 名称 SSV 属性 - InhibitTask SSV 源 - 设置为 1 的 DINT 变量
SFC39 EN_IRT	启用由指定的 OB 处理的中断	SSV 启用指定的任务。	SSV 类 - Task SSV 实例 - Task 名称 SSV 属性 - InhibitTask SSV 源 - 设置为 0 的 DINT 变量
SFC41 DIS_AIRT	禁用由指定的 OB 处理的中断。 中断请求延迟。	UID	由优先级更高的任务禁用当前任务的中断
SFC42 EN_AIRT	启用由指定的 OB 处理的中断。执行由 SFC41 延迟的所有中断。	UIE	启用当前任务的中断。

错误

这些系统调用返回表示错误代码的位字段 （在 S7 中）或整数（在 Logix 中）。

S7	备注	Logix	备注
SFC38 READ_ERR	读取并清除错误位。可以使用筛选字段选择要查询的错误类型。	GSV （使用 SSV 复位计数器或故障）	GSV 类 - FaultLog GSV 属性： MajorEvents - 主要事件数 MinorEvents - 次要事件数 MajorFaultBits - 当前主要故障 MinorFaultBits - 当前次要故障 GSV 目标 - 用于接收数据的 INT 或 DINT

状态 - 控制器

SFC (S7) 和 GSV 调用 (Logix) 将返回控制器的数据。注意：在使用 SFC51 之前，需要进行一定程度的学习。因此，GSV 更易于入手。

S7	备注	Logix	备注
SFC51 RDSYSST	输入参数指定要读取的信息类别，在有多个对象时也可以指定实例号。 输出参数为指向包含返回信息的列表的指针，以及列表中元素的数目和大小。	GSV	具有直接连接的模块：检查“Fault”或“ChannelFault”成员 （如果有）。具有最佳机架式连接的模块：检查适配器输入数据的“SlotStatusBits”成员或卡的“'Fault”成员 （如上）。对于所有其他卡：执行 GSV： 类 - Module 实例 - ModuleName 属性 - Entrystatus

状态 - 模块

SFC (S7) 和 GSV 调用 (Logix) 将返回已安装模块的数据。

S7	备注	Logix	备注
SFC51 RDSYSST	输入参数指定要读取的信息类别，在有多个对象时也可以指定实例号。 输出参数为指向包含返回信息的列表的指针，以及列表中元素的数目和大小。	GSV	GSV 类 - Module GSV 属性： EntryStatus （Module 对象与 Module 之间的关系） FaultCode FaultInfo ForceStatus LEDStatus Mode （同 SSV） GSV 目标 - 取决于所选择的属性

可以监控 Logix 标签中的故障信息，这些标签是在模块插入到 I/O 配置中时创建的。与 STEP 7 类似，如果转到硬件配置并切换到“在线打开”，则会显示模块的故障信息。

状态 - OB 和任务

S7	备注	Logix	备注
OB Header	OB 的状态数据存储在由 OB 头自动生成的临时变量中。这些变量可以由 OB 代码直接访问，并可以在需要从 OB 之外进行访问时传输到静态数据区。 请参见下面的示例。	GSV / SSV	GSV 类 - Task GSV 实例 - Task 名称 GSV 属性： DisableUpdateOutputs （在 Task 末尾） EnableTimeOut InhibitTask Instance LastScanTime （微秒） MaxIntervaln （Task 持续执行之间） OverlapCount （在执行时触发） Priority Rate （以微秒为单位的周期） StartTime （上次启动任务时的 WallClockTime 值） Status （3 个状态位） Watchdog （微秒） GSV 源 / 目标 - 取决于所选择的属性

定时器

S7	备注	Logix	备注
SFB4 TON	接通延时定时器	TON (LD) TONR （ST 和 FBD）	接通延时定时器
		RTO (LD) RTOR （LD 和 ST）	保持型接通延时定时器
SFB5 TOF	关断延时定时器	TOF (LD) TOFR （ST 和 FBD）	关断延时定时器
SFB3 TP	生成一个将无条件运行的脉冲	自由运行的 TON 的累加器的位	

转换例程

S7	备注	Logix	备注
库功能		指令	
FC16 I_STRNG	整数到字符串	DTOS	INT 可代替 DINT 用作源标签
FC5 DI_STRNG	双精度整数到字符串	DTOS	DINT 到字符串
FC30 R_STRG	实数到字符串	RTOS	实数到字符串
FC38 STRG_I	字符串到整数	DTOS	
FC37 STRG_DI	字符串到双精度整数	STOD	字符串到 DINT
FC39 STRG_R	字符串到实数	STOR	字符串到实数

字符串处理例程

S7	备注	Logix	备注
	库功能		指令
FC10 EQ_STRNG	比较字符串是否相等	EQU	比较字符串是否相等
FC13 GE_STRNG	比较一个字符串是否大于等于另一个字符串	GEQ (LD) >= (ST)	比较一个字符串是否大于等于另一个字符串
FC15 GT_STRNG	比较一个字符串是否大于另一个字符串	GRT (LD)	比较一个字符串是否大于另一个字符串
FC19 LE_STRNG	比较一个字符串是否小于等于另一个字符串	LEQ (LD) <= (ST)	比较一个字符串是否小于等于另一个字符串
FC24 LT_STRNG	比较一个字符串是否小于另一个字符串	LES (LD) < (ST)	比较一个字符串是否小于另一个字符串
FC29 NE_STRNG	比较字符串是否不相等	NEQ (LD) <> (ST)	比较字符串是否不相等
FC21 LEN	字符串的长度	.LEN	所有字符串实例的属性
FC26 MID	返回字符串的中间部分	MID	返回字符串的中间部分
FC2 CONCAT	串联两个字符串	CONCAT	串联两个字符串
	可以使用 FC31 REPLACE 实现	DELETE	删除字符串的某个部分
FC17 INSERT	将源字符串插入到目标字符串中	INSERT	将源字符串插入到目标字符串中
FC31 REPLACE	使用源字符串替换目标字符串的 n 个字符	使用 DELETE / INSERT	
FC11 FIND	在字符串中查找另一个字符串	FIND	在字符串中查找另一个字符串

在 STEP 7 中，没有 Logix 的 ASCII 串口指令的等效项？ 无论是在指令集中还是在功能库中。如果需要这些功能，必须在 STL 中进行编程。

系统功能调用的示例

这些示例主要用于演示 GSV/SSV 指令的用法。

设置时钟

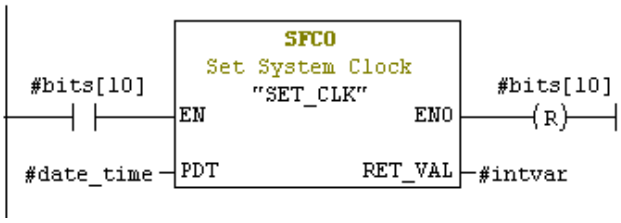
STEP 7

对 SFC0 的这一调用会设置时钟。在 #date_time 中输入时间和日期。

日期和时间以 BCD 格式存储在 #data_time 后的 8 个字节中。

Network 14 : Title:

```
set the clock to the value stored in "date_time"
```



0 - 年

1 - 月

2 - 日

3 - 小时

4 - 分钟

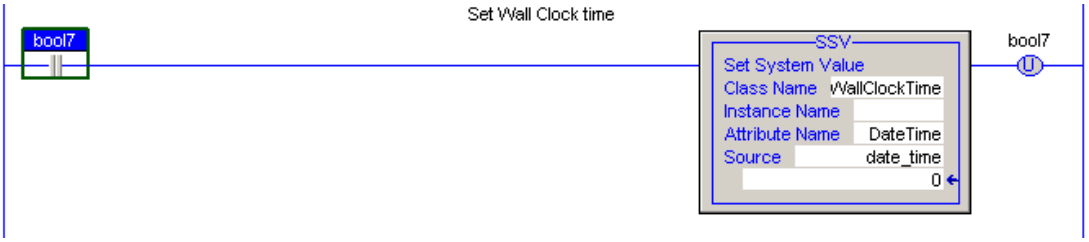
5 - 秒

6 - 毫秒的最高 2 位有效数字

7 - 毫秒的最低 1 位有效数字和星期几

Logix

日期和时间值存储在 #date_time 后的七个 DINT 中。



0 - 年

1 - 月

2 - 日

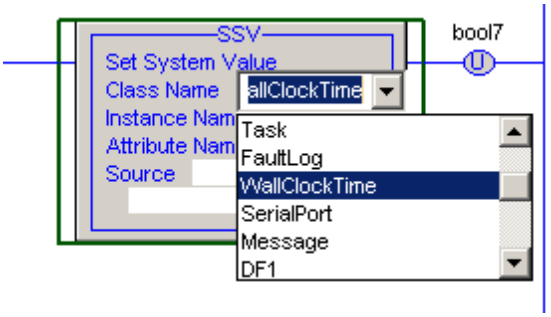
3 - 小时

6 - 分钟

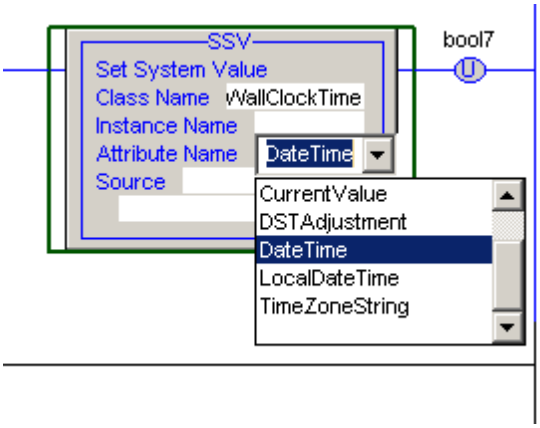
5 - 秒

6 - 微秒

Logix 的屏幕截图显示与 GSV 和 SSV 关联的数据结构。从下拉菜单中选择类别，如下所示。



从下拉菜单中选择属性，如下所示。



最后，选择将作为数据的源（SSV）或目标（GSV）的标签。

禁用中断

STEP 7

Network 2 : Title:

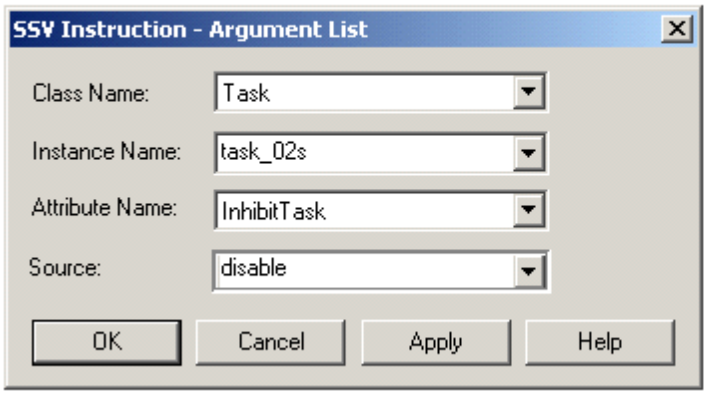
```
Disable interrupts for the Interrupt Execution (ie Periodic) OB35

CALL  "DIS_IRT"                SFC39          -- Disable New Inte
 MODE   :=B#16#2
 OB_NR  :=35
 RET VAL:=#intVar
```

Logix

此示例以结构文本演示 SSV。

如果键入 “gsv”，然后按 “alt-A”，则会弹出下面的参数选择屏幕。

```
// disable task_0.2s
ssv()
```

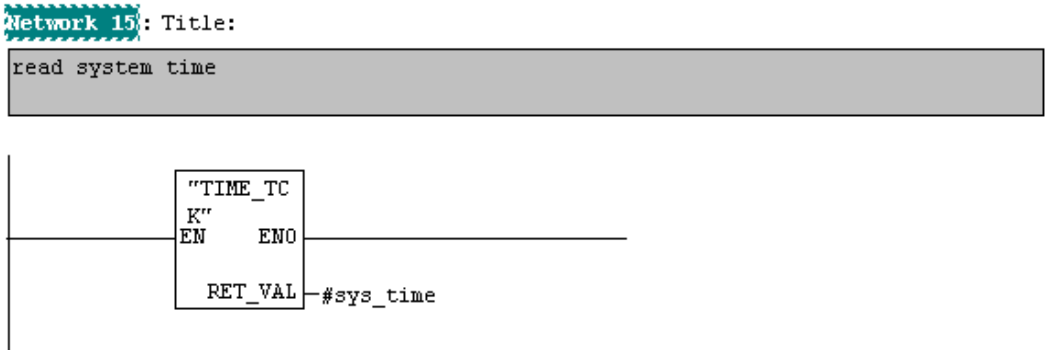


在输入参数后，单击 “OK”，即完成实际参数设置。

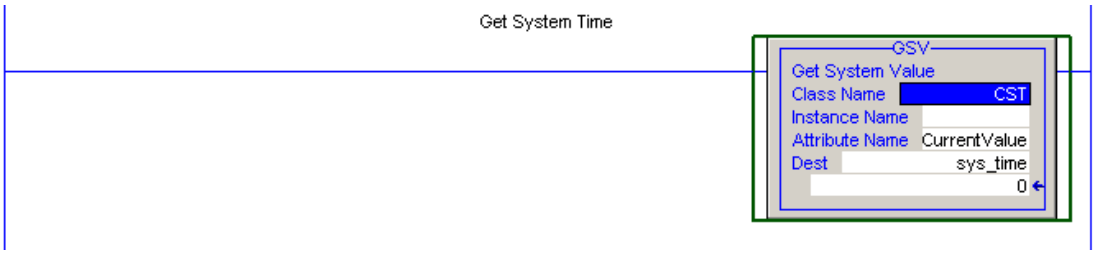
```
// disable task_0.2s
ssv(Task,task_02s,InhibitTask,disable);
```

读取系统时间

STEP 7



Logix

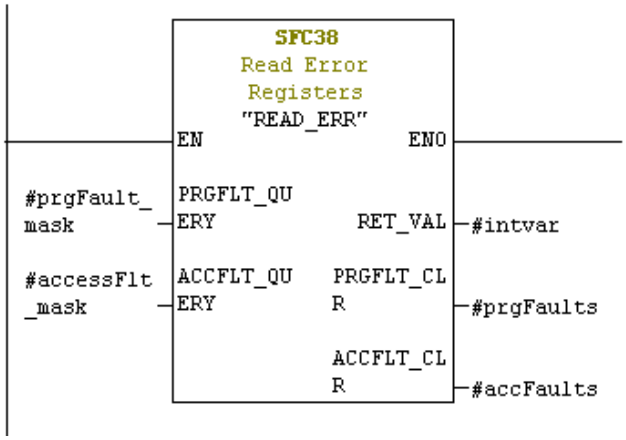


获取故障

STEP 7

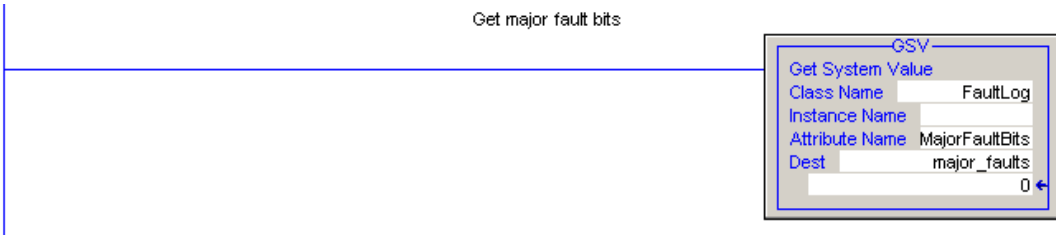
Network 16 : Title:

```
Get programming faults and I/O access faults
```



输入参数中的位模式用作筛选器，以选择要查询的故障。返回的故障为**屏蔽后**的故障 – 屏蔽可防止这些故障停止控制器或调用故障 OB。

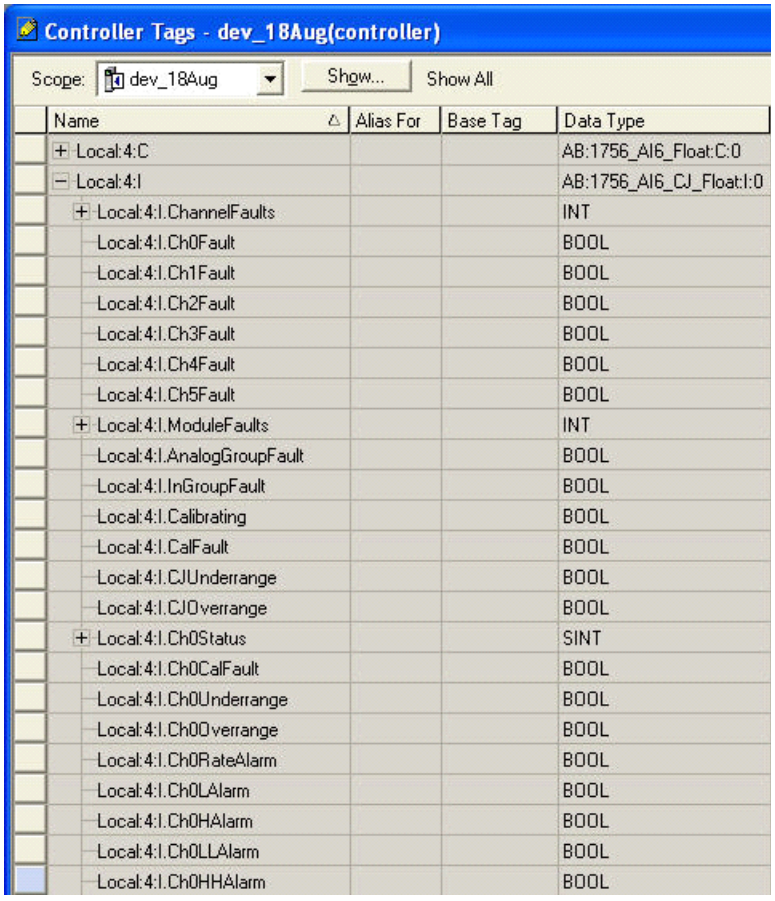
Logix



模块信息

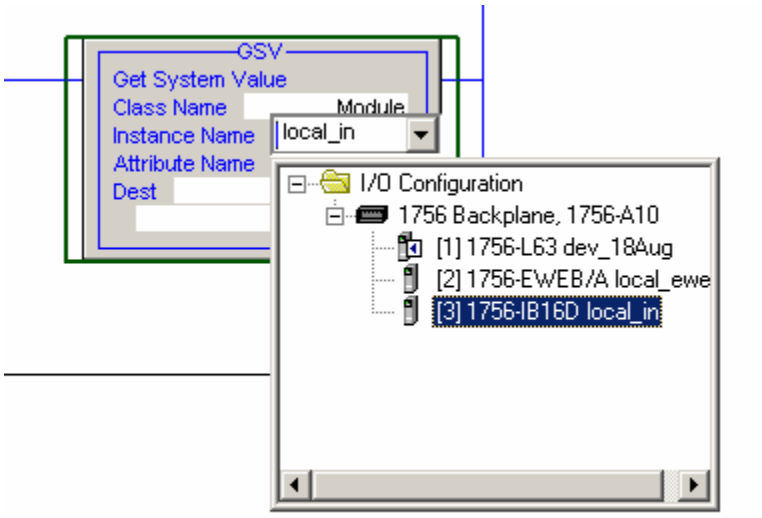
最简单的方式是检查模块设备配置文件标签，这些标签包含故障 / 诊断信息。

1756-IT6I2 热电偶模拟输入卡标签



Name	Alias For	Base Tag	Data Type
Local:4:C			AB:1756_AI6_Float:C:0
Local:4:I			AB:1756_AI6_CJ_Float:I:0
Local:4:I.ChannelFaults			INT
Local:4:I.Ch0Fault			BOOL
Local:4:I.Ch1Fault			BOOL
Local:4:I.Ch2Fault			BOOL
Local:4:I.Ch3Fault			BOOL
Local:4:I.Ch4Fault			BOOL
Local:4:I.Ch5Fault			BOOL
Local:4:I.ModuleFaults			INT
Local:4:I.AnalogGroupFault			BOOL
Local:4:I.InGroupFault			BOOL
Local:4:I.Calibrating			BOOL
Local:4:I.CalFault			BOOL
Local:4:I.CJUnderrange			BOOL
Local:4:I.CJOverrange			BOOL
Local:4:I.Ch0Status			SINT
Local:4:I.Ch0CalFault			BOOL
Local:4:I.Ch0Underrange			BOOL
Local:4:I.Ch0Overrange			BOOL
Local:4:I.Ch0RateAlarm			BOOL
Local:4:I.Ch0LAlarm			BOOL
Local:4:I.Ch0HAlarm			BOOL
Local:4:I.Ch0LLAlarm			BOOL
Local:4:I.Ch0HHAlarm			BOOL

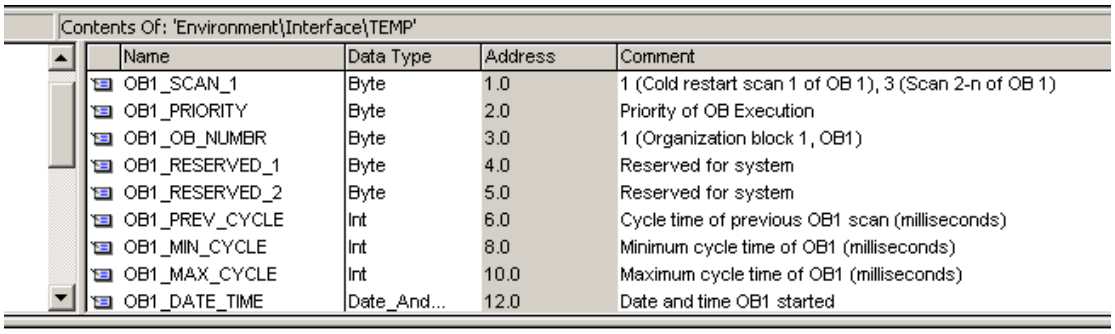
另一种方式是使用 GSV 指令读取模块对象。下面的屏幕截图显示如何使用 GSV 获取 1756-IB16D 数字输入模块的相关信息。



获取扫描时间

STEP 7

这是 OB1 的临时变量头的屏幕截图。


Contents Of: 'Environment\Interface\TEMP'

Name	Data Type	Address	Comment
OB1_SCAN_1	Byte	1.0	1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
OB1_PRIORITY	Byte	2.0	Priority of OB Execution
OB1_OB_NUMBR	Byte	3.0	1 (Organization block 1, OB1)
OB1_RESERVED_1	Byte	4.0	Reserved for system
OB1_RESERVED_2	Byte	5.0	Reserved for system
OB1_PREV_CYCLE	Int	6.0	Cycle time of previous OB1 scan (milliseconds)
OB1_MIN_CYCLE	Int	8.0	Minimum cycle time of OB1 (milliseconds)
OB1_MAX_CYCLE	Int	10.0	Maximum cycle time of OB1 (milliseconds)
OB1_DATE_TIME	Date_And...	12.0	Date and time OB1 started

#OB1_PREV_CYCLE 为扫描时间。作为临时变量，在执行完 OB1 之后，它就不再存在。若要存储扫描时间，请将 #OB1_PREV_CYCLE 复制到静态内存位置。

Logix

可以为每个 Logix 任务检索执行时间。

使用 S7 时，可以直接从 #OB1_PREV_CYCLE 获取 OB1 的扫描时间。但是，对于周期性 OB，不存在 #OB1_PREV_CYCLE 的等效项。若要获取周期性 OB 的执行时间，需要在 OB 的开始和结束处插入对 SFC64 TIME_TCK 的调用，并减去由 SFC 返回的系统时钟时间。

典型程序结构的转换

简介

本节的目的是说明如何在 RSLogix 5000 软件中执行 STEP 7 中的一些典型编程任务。讨论主要以代码段为基础，但是也提供一些完整的示例。

此外，还将讨论一些与编程有关的问题，如变量的作用域和可见性，以及代码段的安排。

转换代码示例

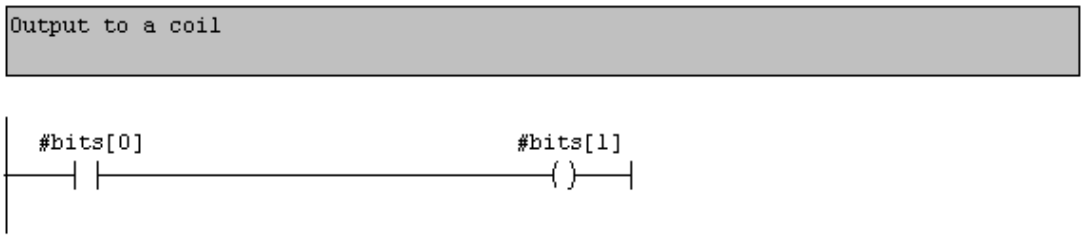
这些示例演示转换代码。

梯形图逻辑转换

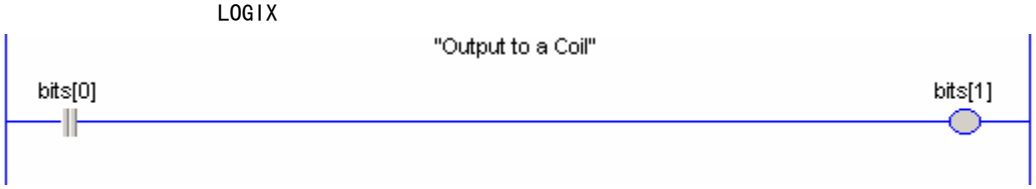
本节介绍一些对 STEP 7 LAD 与 Logix LD 进行比较的示例。

写入一位

STEP 7

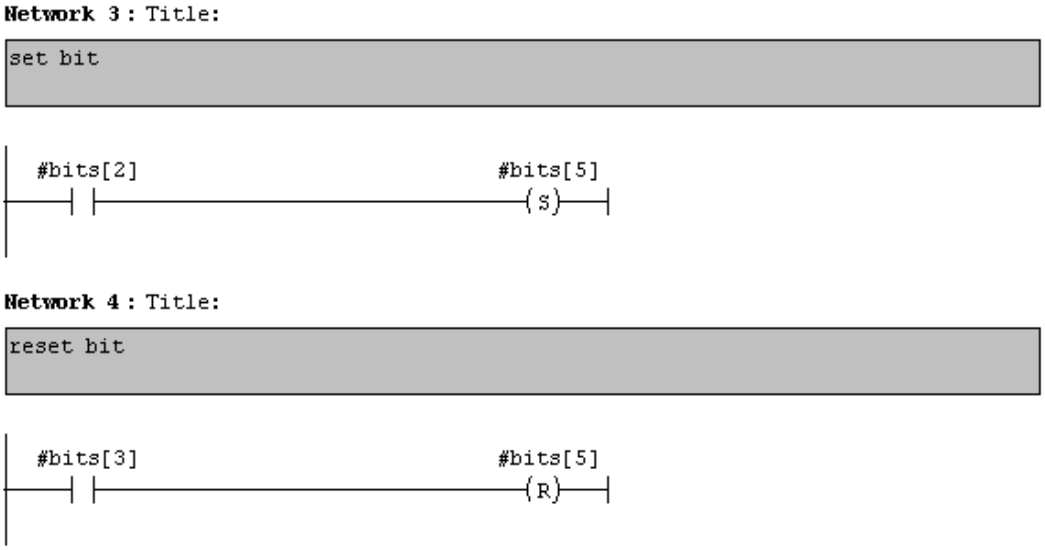


LOGIX



设置和复位

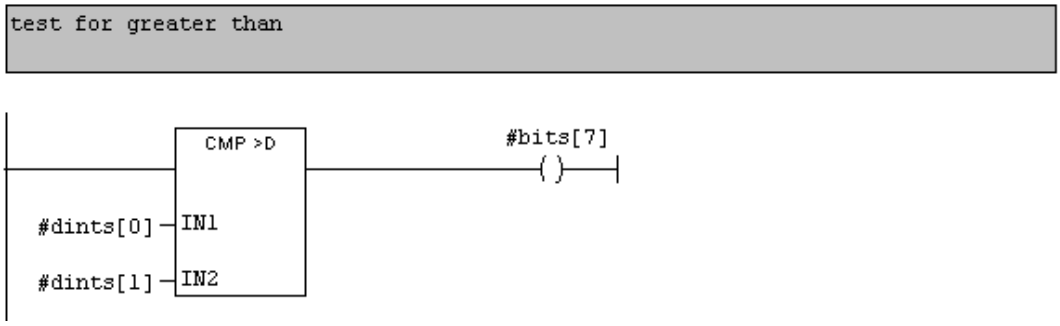
STEP 7



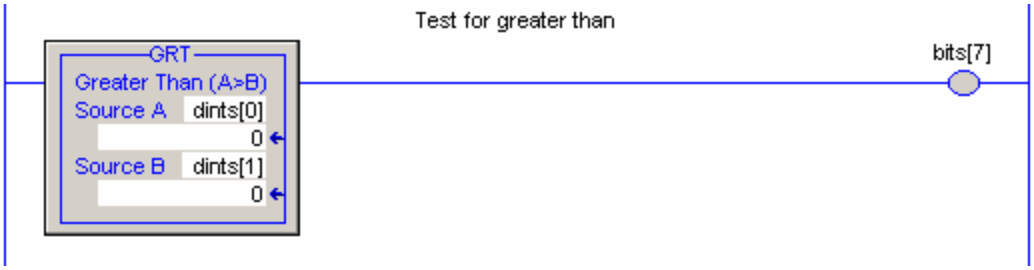
LOGIX

测试是否大于

STEP 7



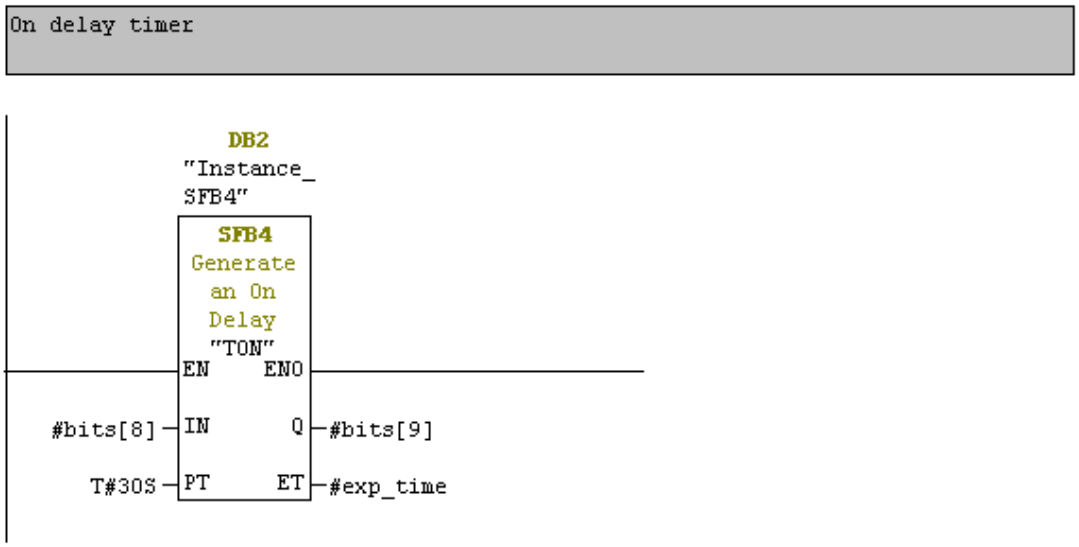
LOGIX



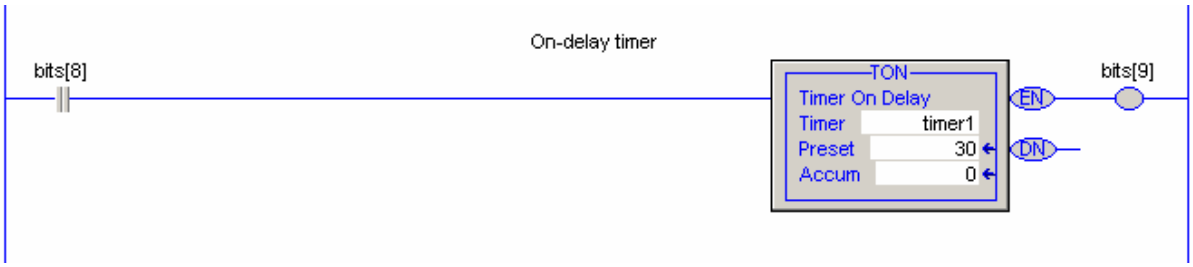
与以前一样，如果表达式不是仅仅比较两个数这么简单，请使用 CMP 指令。

接通延迟定时器

STEP 7

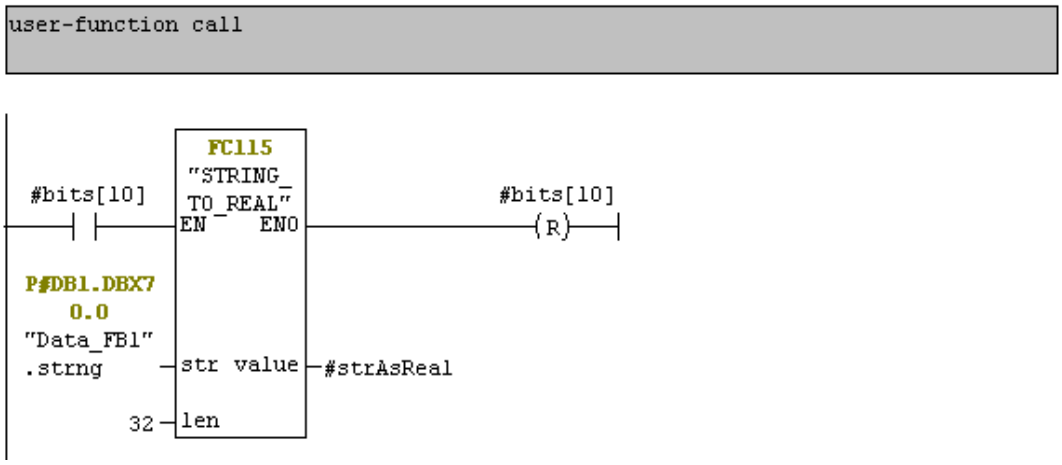


LOGIX

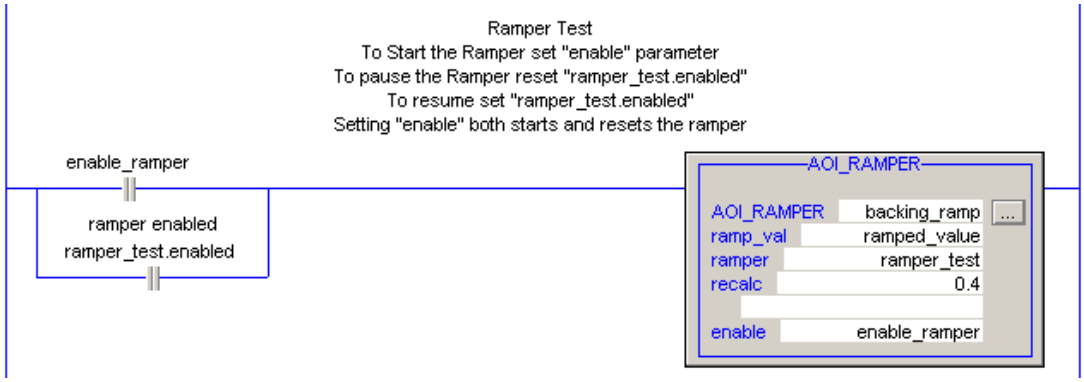


调用用户功能

STEP 7

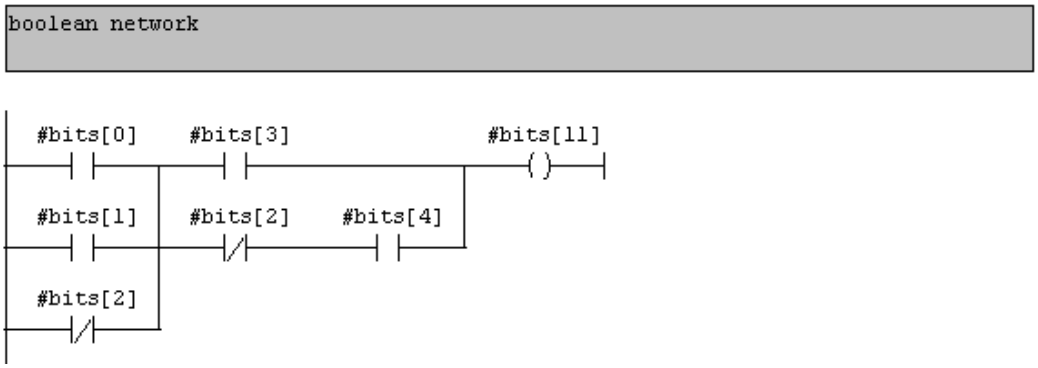


LOGIX

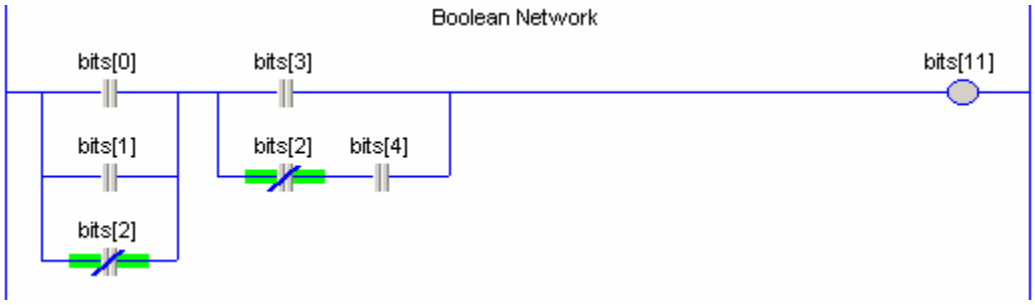


布尔网络

STEP 7



LOGIX

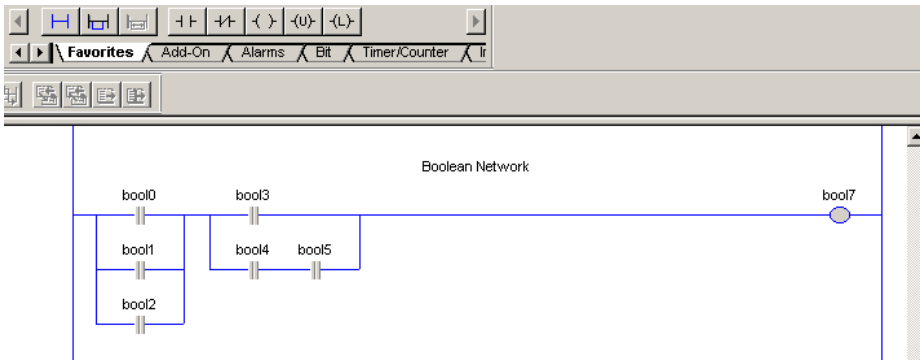


STEP 7 LAD 与 Logix LD 的相似程度很高，可以相当容易地在例程级别进行转换。

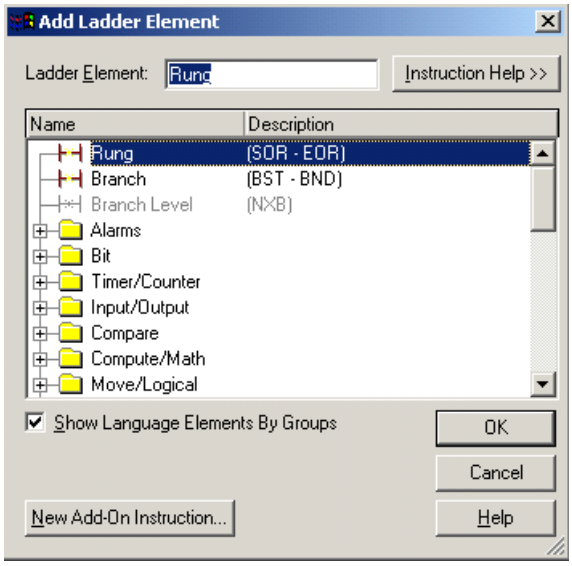
Logix LD 编辑器

至少有七种方法可用于选择 LD 指令。下面介绍的这两种方法与 STEP 7 中的方法十分相似。

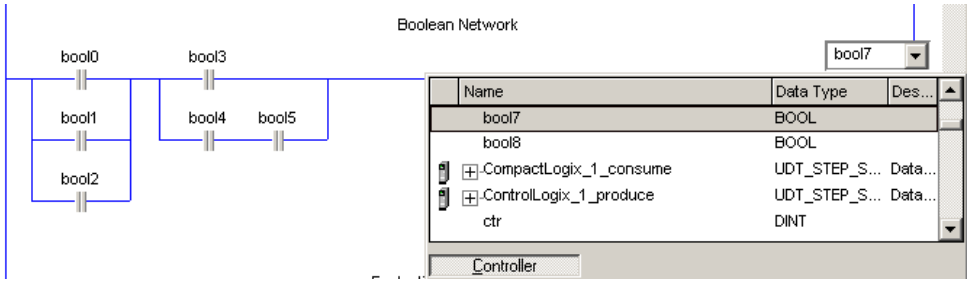
可以从 LD 工作表上的调板进行选择。



如果按下 Alt+Insert，则会显示此选择弹出菜单。



在配置指令时，可以使用下拉菜单选择要输入的标签。



转移和决策

STEP 7 - 传统转移序列

下面的示例任务在网络注释中进行了说明。所显示的是两个 S7 版本，因为这两个版本都经常用到。

```
Network 1: Multi-way selection
if #input is 5 set #target to 8

else if #input is 6 set #target to 10

else if #input is 7 set #target to 16

else set #input to 0
```

```
      L     #input
      L     5
      ==I
      JCN   _001

      L     8
      T     #target

      JU    end

_001: L     #input
      L     6
      ==I
      JCN   _002

      L     10
      T     #target

      JU    end

_002: L     #input
      L     7
      ==I
      JCN   _003

      L     16
      T     #target

      JU    end

_003: L     0
      T     #target

end:  NOP   0
```

将 #input 的值与常数集进行比较，直至找到匹配项为止。然后执行操作并停止比较。如果 #input 与集中的任何值都不匹配，则执行默认操作。

STEP 7 - 转移列表

在此示例中，还是同样的任务，但使用了转移列表。这与微处理器转移表相似，根据变量的值将执行传递给标签。

Network 2 : Title:

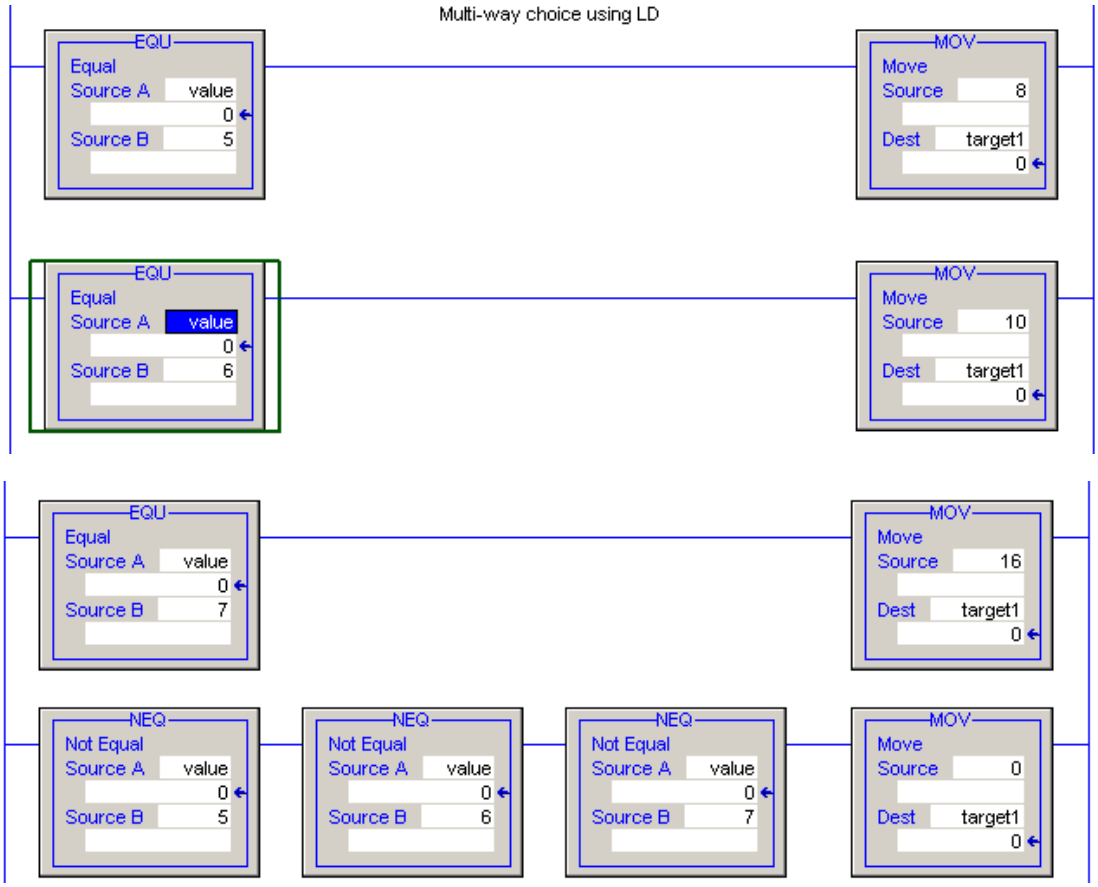
```
if #input is 5 set #target to 8

else if #input is 6 set #target to 10

else if #input is 7 set #target to 16

else set #input to 0
```

```
          L     #input
          L     5
          -I
          JL    rng
          JU    d5
          JU    d6
          JU    d7
    rng:  L     0
          T     #target
          JU    cont

    d5:   L     8
          T     #target
          JU    cont

    d6:   L     10
          T     #target
          JU    cont

    d7:   L     16
          T     #target

    cont: NOP   0
```

这比传统转移序列的可读性更强，并且更加高效，这时因为仅执行目标标签处的代码。

Logix - 梯形图逻辑

下图演示使用 LD 的多分支选择。



Logix - 结构文本 If…Then…Else

任何熟悉 Basic/Pascal/C 系列编程语言的人都很容易理解这种结构文本。

```
// multi-way choice using Structured Text

if (value = 5) then target := 8;
elsif (value = 6) then target := 10;
elsif (value = 7) then target := 16;
else target := 0;
end_if;
```

“if”条件周围的括号不是必需的。

Logix 结构文本 CASE 语句

这是 ST 中执行相同任务的另一种变体。它简洁清晰，几乎不需要附加注释。

```
// multi-way choice using Structured Text CASE

case value of
    5: target := 8;
    6: target := 10;
    7: target := 16;
else target :=0;
end_case;
```

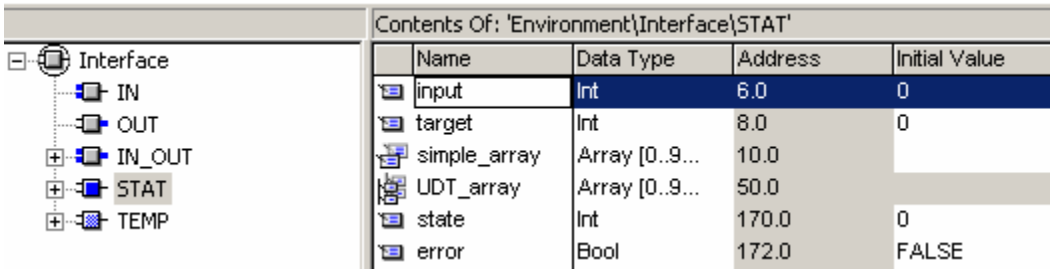
所有解决方案都适用，但这是首选的 Logix 解决方案。这种方案十分简洁并且足够清晰，不需要进一步说明。

数组

STEP 7 和 Logix 都允许在内存中创建简单或复杂对象的数组。Logix 为存取数组提供高级支持。但是在 STEP 7 中需要进行低级编程。

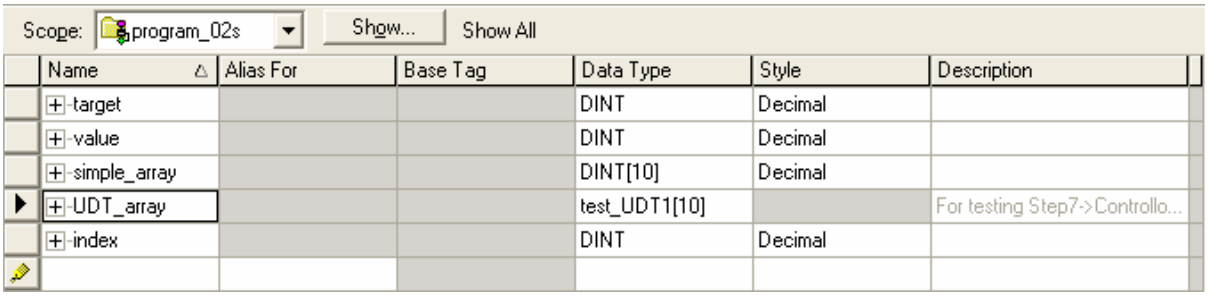
STEP 7 数组创建

下面的屏幕截图显示在实例数据块中创建的两个数组。Simple_array 是 10 个元素组成的数组。UDT_array 是 10 个类型为 test_UDT1 的结构组成的数组，其中 test_UDT1 是包含其他类型 （未显示）的用户数据类型。



Logix 数组创建

在 Logix 中，情况完全相同。



数组声明语法

STEP 7 使用声明语法 ARRAY[0…15] OF REAL。Logix 使用 REAL[15]。

STEP 7 有一条特殊的字符串语法。在 STEP 7 中，STRING[32] 是包含 32 个字符的字符串，而在 Logix 中，STRING[32] 是包含 32 个字符串的数组，其中每个字符串包含 82 个字符。

STEP 7 中的数组存取

此示例将对 simple_array[] 和 UDT_array[] 这两个数组执行一个简单任务。该任务在网络注释中进行了说明。

在 STEP 7 中，不可能使用常规 array[] 表示法存取数组。而必须对指针使用低级操作。在下面的代码段中，功能“GET_INDEXED_REFERENCE”通过返回要存取的数组元素的指针，简化了任务。

Network 3 : Title:

```
array operations
----------------
if (simple_array[2] = simple_array[5]) then
  UDT_array[8].booleanl := 1;
else
  UDT_array[8].booleanl := 0;
end_if;
```

```
    // 1.  compare simple_array[2] with simple_array[5]
          CALL  "GET_INDEXED_REFERENCE"            FC111
           refArray  :="Data_test".simple_array    P#DB1.DBX10.0
           index     :=2
           byteIncr  :=4
           startIndex:=FALSE
           retVal    :=#ptr1                              BOOL

          CALL  "GET_INDEXED_REFERENCE"            FC111
           refArray  :="Data_test".simple_array    P#DB1.DBX10.0
           index     :=5
           byteIncr  :=4
           startIndex:=FALSE
           retVal    :=#ptr2

          OPN   "Data_test"                        DB1
          L     DID [#ptr1]
          L     DID [#ptr2]
          ==D
          =     #compare

   // 2. get pointer to UDT_array
          CALL  "GET_INDEXED_REFERENCE"            FC111
           refArray  :="Data_test".UDT_array       P#DB1.DBX44.0
           index     :=8
           byteIncr  :=12     INT
           startIndex:=FALSE
           retVal    :=#ptr1

          L     #ptr1
          LAR1

   // 3. set or reset the bit
          A     #compare
          =     DIX [AR1,P#0.0]
```

在此示例中，将实际 Logix 结构文本代码用作网络注释，以此说明 Logix 代码的直观性。

STEP 7 - 遍历数组元素

此示例的目标是清除 UDT_array[] 中所有结构中的浮点字段。这并不困难，但需要熟练使用指针。

```
Array Operations
----------------
Clear all float elements at UDT offset P#6.0 in array UDT_array
```

```
// transfer pointer to UDT_array to AR1
      L     P##UDT_array
      LAR1
// initialise counter
      L     0
      T     #ctr

// end if #ctr > 9
loop: L     #ctr
      L     9
      >I
      JC    end2
// clear the float field at offset p#6.0
      L     0.000000e+000
      T     DID [AR1,P#6.0]
// increment AR1 by size of the UDT
      +AR1  P#12.0
// increment counter
      L     #ctr
      INC   1
      T     #ctr
// loop back
      JU    loop

end2: NOP   0
```

Logix - 结构文本中的数组操作

下面的 ST 段执行在前两节中介绍的任务。

```
// array access in ST
if (simple_array[2] = simple_array[5]) then
    UDT_array[8].booleanl := 1;
else
    UDT_array[8].booleanl := 0;
end_if;


// clearing array elements
if (simple_array[0] = 5) then
    index := 0;
    while (index <= 9) do
        UDT_array[index].reall := 0.0;
        index := index + 1;
    end_while;
end_if;
```

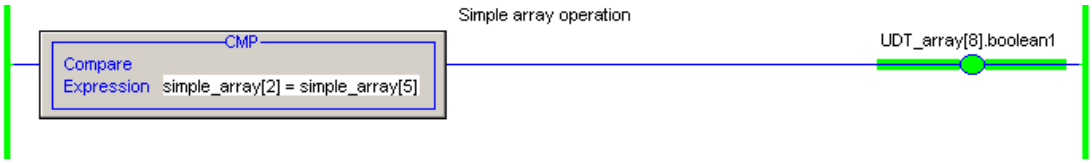
这段代码非常清晰，无需附加任何注释。

如果要用 if…then…else 语句切换布尔变量，可考虑改为编写布尔等式：

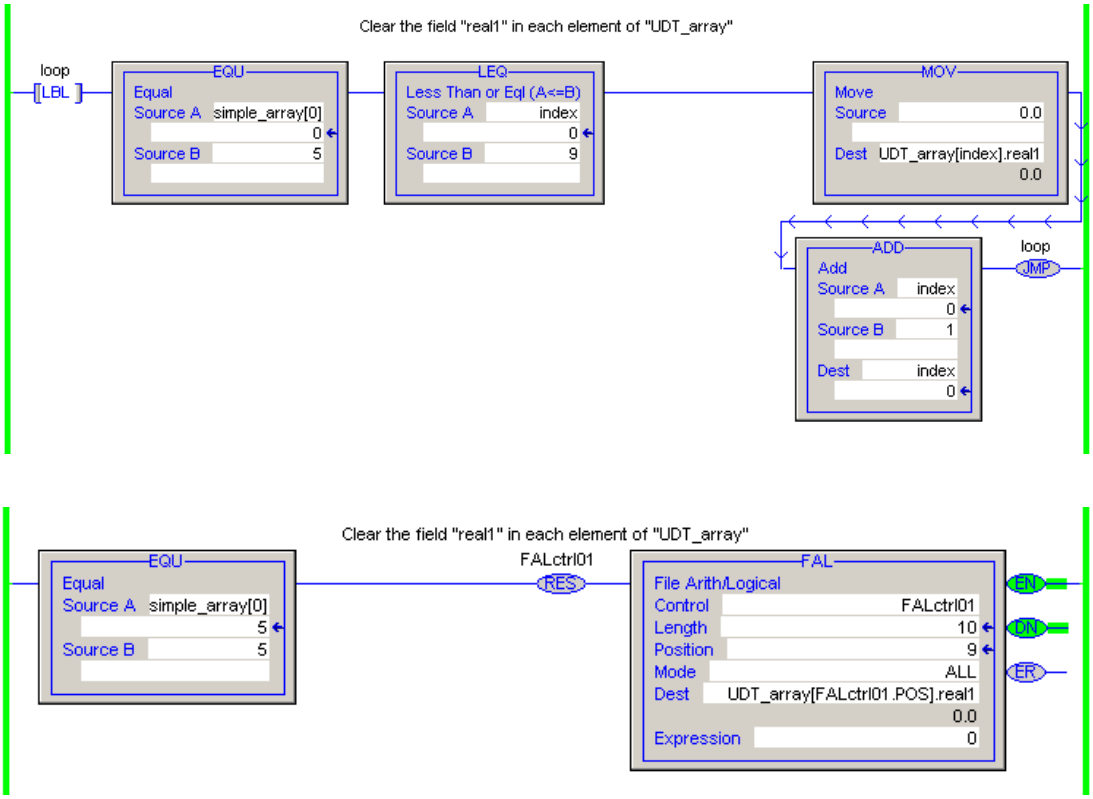
```
// array access in ST
UDT_array[8].booleanl := simple_array[2] = simple_array[5];
```

Logix - 梯形图中的数组操作

上节中的示例可以在 LD 中使用 CMP （比较）指令编写，如下所示。



这两种方式都可以完成第二个示例 （清除 UDT 数组中的实数字段）。



清除数组元素的第一种方法是从 ST 代码的 While 循环转换而来。第二种方法将高级 FAL 指令用于数组操作。

用户数据类型

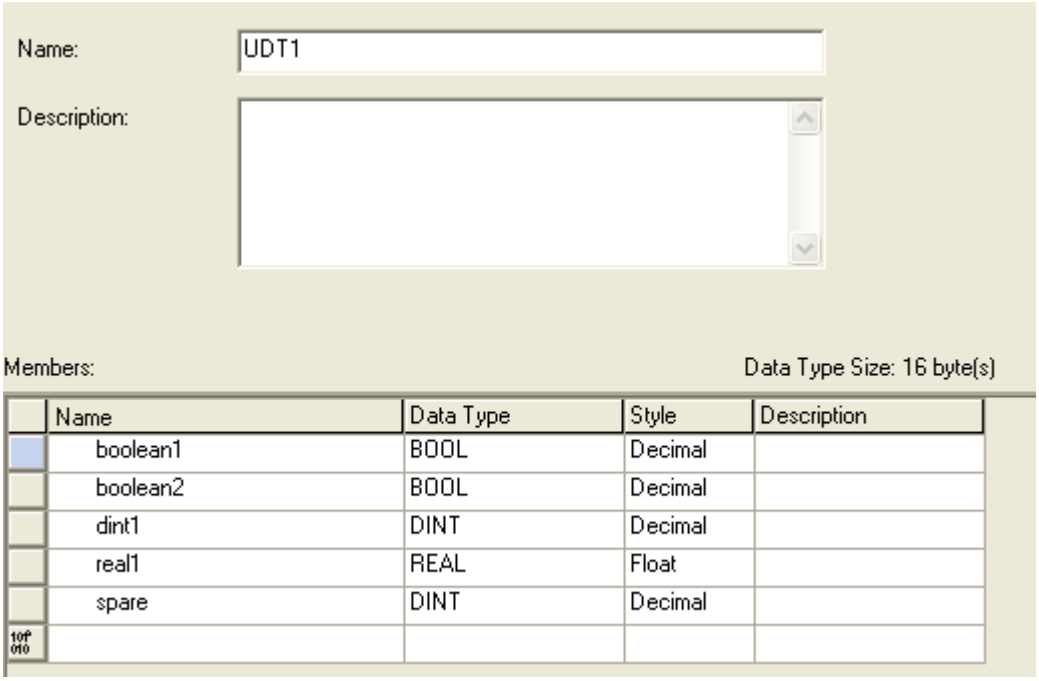
在 STEP 7 和 Logix 中，用户数据类型 (UDT) 的配置和使用非常相似。

下面是 STEP 7 中的一个 UDT。

UDT1 -- Conversion_examples\SIMATIC 400 Station\CPU414-2DP(1)

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	boolean1	BOOL	FALSE	
+0.1	boolean2	BOOL	FALSE	
+2.0	dint1	DINT	L#0	
+6.0	real1	REAL	0.000000e+000	
+10.0	spare	WORD	W#16#0	
=12.0		END_STRUCT		

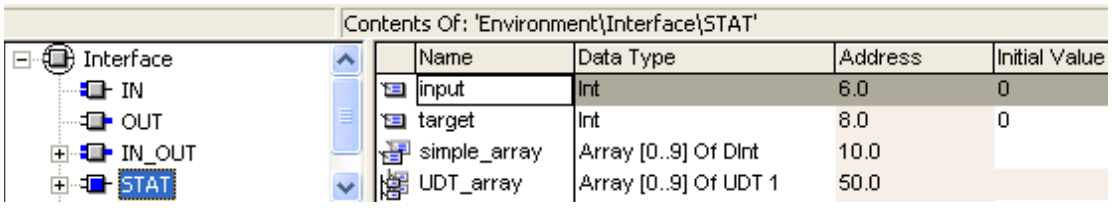
下面是 Logix 中的一个 UDT。



Name	Data Type	Style	Description
boolean1	BOOL	Decimal	
boolean2	BOOL	Decimal	
dint1	DINT	Decimal	
real1	REAL	Float	
spare	DINT	Decimal	

在两种系统中，都可以使用 UDT 声明和定义变量。

下面是 STEP 7 中一个包含 UDT 的声明。

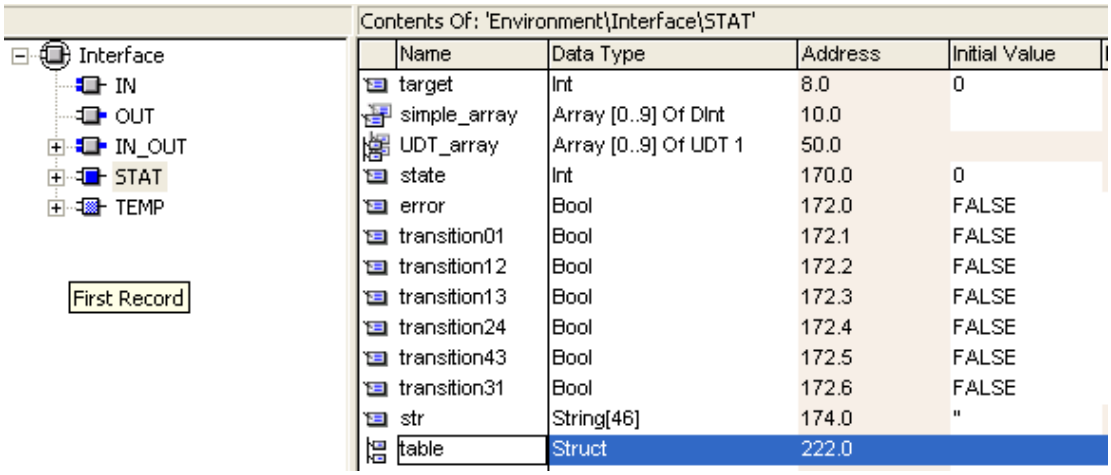


下面是 Logix 中一个包含 UDT 的声明。

Name	Alias For	Base Tag	Data Type	Style
Limit_Switch_1	Local:3:I.Data.0	Local:3:I.Data.0	BOOL	Decimal
Local:3:C			AB:1756_DI:C:0	
Local:3:I			AB:1756_DI:I:0	
conveyor_1			UDT1	
conveyor_1.boolean1			BOOL	Decimal
conveyor_1.boolean2			BOOL	Decimal
conveyor_1.dint1			DINT	Decimal
conveyor_1.real1			REAL	Float
conveyor_1.spare			DINT	Decimal

这两种系统的一个细微差别如下：

在 STEP 7 中，可以声明类型为 “struct”的变量。



注意 Struct 类型的条目 “table”。“table”内可以是包含任何类型组合的集合（有序或无序）。

在 Logix 中，可通过以下方式实现：将 “Struct”定义为包含所需数据结构的 UDT，然后将 “table”声明为类型 Struct。

Name	Alias For	Base Tag	Data Type	Style
Limit_Switch_1	Local:3:I.Data.0	Local:3:I.Data.0	BOOL	Decimal
Local:3:C			AB:1756_DI:C:0	
Local:3:I			AB:1756_DI:I:0	
conveyor_1			UDT1	
conveyor_1.boolean1			BOOL	Decimal
conveyor_1.boolean2			BOOL	Decimal
conveyor_1.dint1			DINT	Decimal
conveyor_1.real1			REAL	Float
conveyor_1.spare			DINT	Decimal
table			Struct	
table.status_bit1			BOOL	Decimal
table.status_bit2			BOOL	Decimal
table.status_bit3			BOOL	Decimal
table.dwell_timer1			TIMER	
table.dwell_timer2			TIMER	
table.speed			REAL	Float

指针和数组

STEP 7 程序可以使用指向任何数据对象的指针。可以对数据块进行间接存取，但是没有指向功能的指针 （除非通过 JL （转移列表）指令这样的严格方式）。数据指针不是普通的指针，因为它指向一个位。因此，它的值是指向普通字节的指针的八倍。这反映出位在控制系统编程中的重要性。

Logix 中没有指针。数组执行的功能与指针相同，但更加简单，也更安全。

S7 程序员是否可以在不使用指针的情况下完成 Logix 中的所有任务？在计算机编程中，数据指针主要用于三个目的：

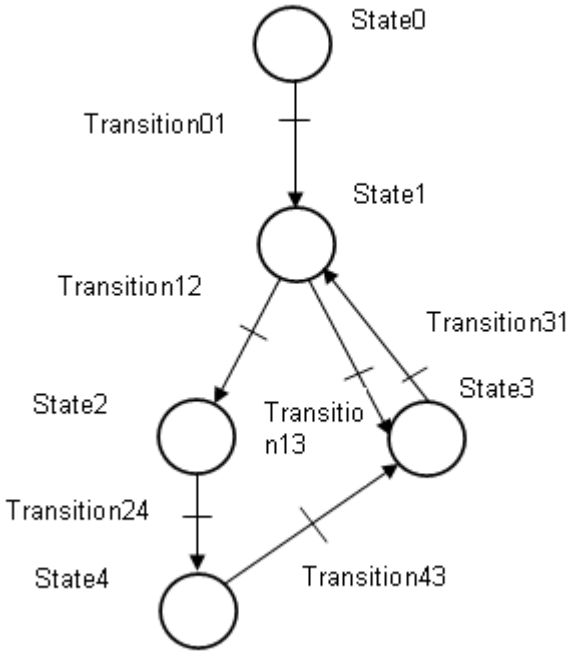
- 对连续排序的数据项 （对象、字符串的数组）进行操作。
- 动态分配、存取和删除已分配的对象。
- 在功能调用中将对对象的引用作为参数进行传递。

在 Logix 中，第一个目的是通过数组实现的。第二个目的与控制软件无关，因为没有动态分配的对象。第三个目的是通过 STEP 7 功能块和 Logix 附加指令中的 “inout”参数实现的。

因此可以得出结论，对于 Logix 程序员而言，缺少显式指针不是一种限制。STEP 7 程序员还会发现，使用数组的代码在使用数组的结构文本中的执行速度比在使用指针的 STL 中的执行速度更快。

状态机器

状态机器是控制系统软件中的重要结构，因为它可极大地简化编程顺序控制的任务。



STEP 7 状态机器

STEP 7 提供图形顺序功能图作为基本应用的可选附加功能。如果图形 SFC 不可用，则语句列表将执行该功能。

Network 4: Title:

state machine

```
          L     #state
          JL    rngl
          JU    st0
          JU    st1
          JU    st2
    rngl: SET
          S     #error
          BEU

    st0:  L     1
          A     #transition01
          JC    next
          JU    ovr

    st1:  L     2
          A     #transition12
          JC    next

          L     3
          A     #transition13
          JC    next
          JU    ovr

    st2:  L     4
          A     #transition24
          JC    next
          JU    ovr

    st3:  L     0
          A     #transition31
          JC    next
          JU    ovr

    st4:  L     3
          A     #transition43
          JC    next
          JU    ovr

    next: T     #state

    ovr:  NOP   0
```

变量 #state 包含状态编号。转移列表指令使执行跳转到与 #state 的值相关的标签。如果来自该状态的转换条件为 True，则新的状态值会加载到累加器中，且执行会跳转到标签 “next”，其中的新状态编号会传送到变量 #state 中。

结构文本中的 Logix 状态机器

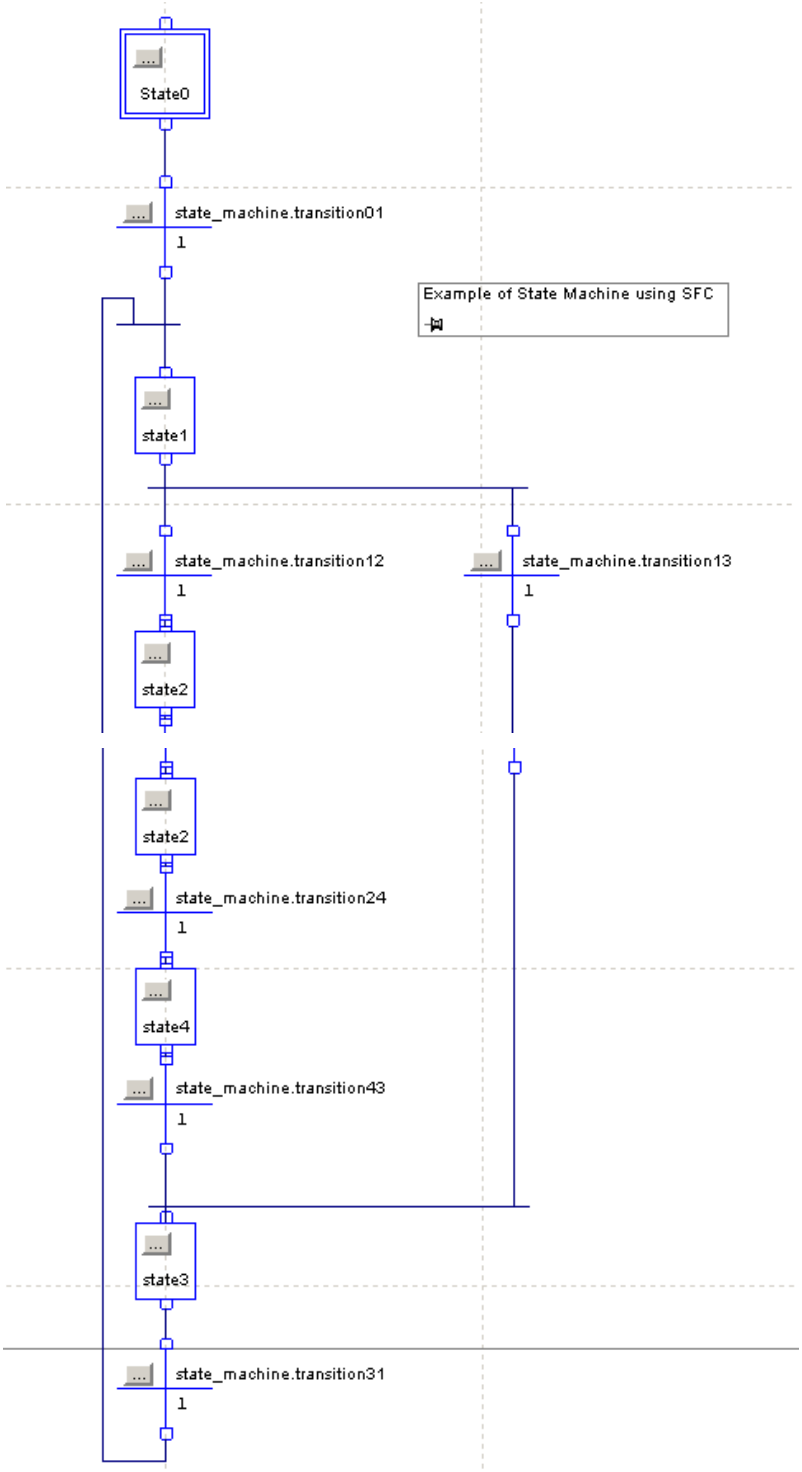
下面是使用 CASE 语句、以结构文本编写的相同状态机器。与其他 ST 示例一样，代码本身就一目了然，无需编写说明。

```
// implementation of State Machine using CASE in ST

case state_machine.state of
    0:  if state_machine.transition01 then
            state_machine.state := 1;
        end_if;
    1:  if state_machine.transition12 then
            state_machine.state := 2;
        elsif state_machine.transition13 then
            state_machine.state := 3;
        end_if;
    2:  if state_machine.transition24 then
            state_machine.state := 4;
        end_if;
    3:  if state_machine.transition31 then
            state_machine.state := 1;
        end_if;
    4:  if state_machine.transition43 then
            state_machine.state := 3;
        end_if;
end_case;
```

顺序功能图中的 Logix 状态机器

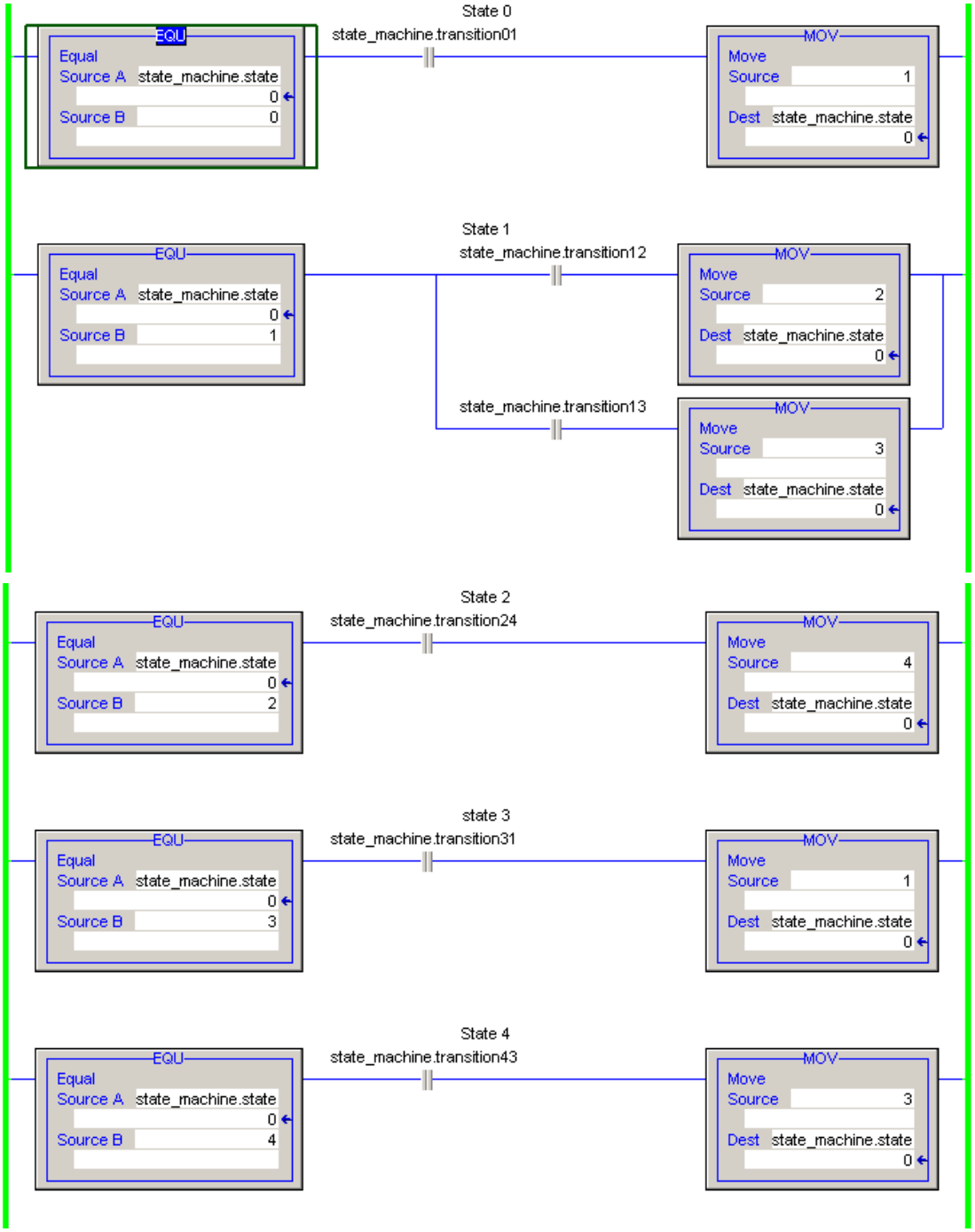
Logix 提供图形 SFC 作为语言的标准套件之一。下面显示的是 SFC 中的状态机器。

使用 SFC 图的状态机器的实施



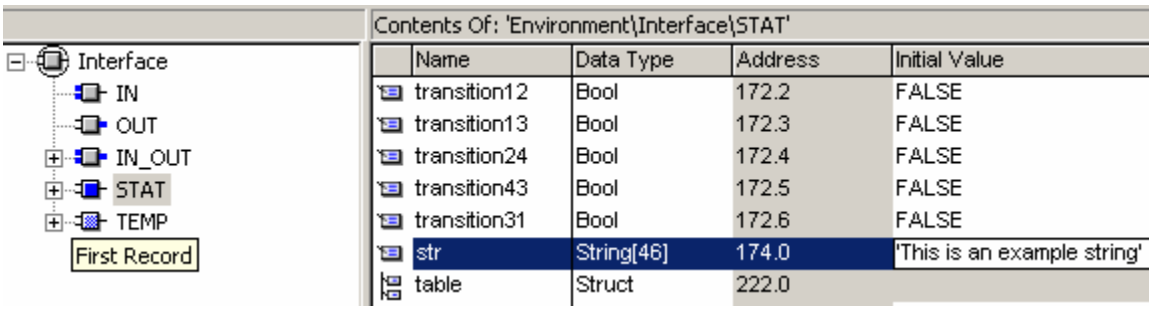
梯形图中的状态机器

下面的屏幕截图演示如何在梯形图中实施状态机器。



字符串

STEP 7 中的字符串定义



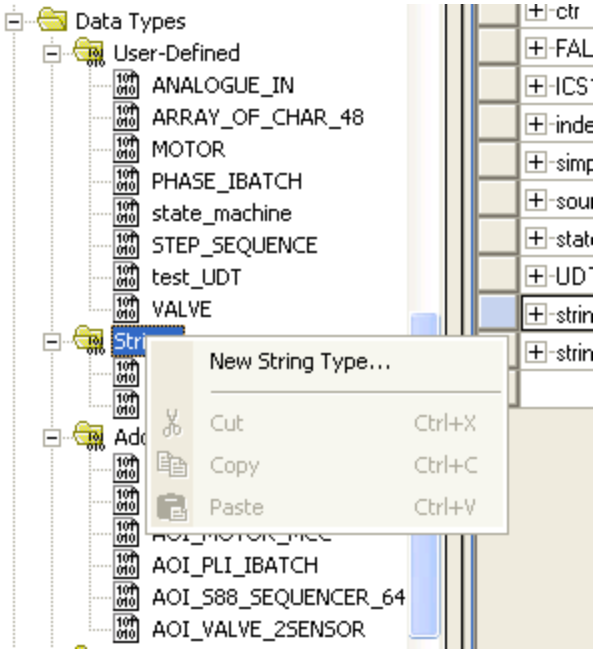
数据标头显示字符串的定义方式。字符串的长度是在 String 数据类型后的方括号 [] 中输入的。字符串初始值是在 “Initial Value” 列中键入的。

可以创建字符串数组，但不能为每个字符串提供初始值。图中显示了可避免这一问题的另一种定义，即数据标头中的条目 “table”。“Table” 是一种结构。该结构的内容 （未显示）是 string[46] 的五个实例，每个实例都提供有一个初始值。

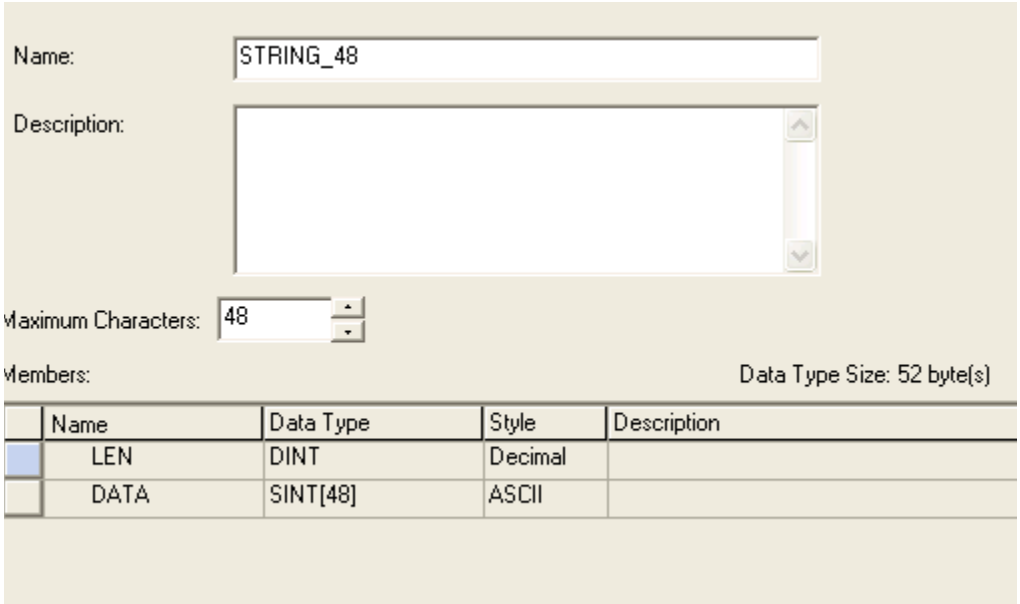
Logix 中的字符串定义

下面的内容摘自标签配置，说明在 Logix 中是如何定义字符串的。

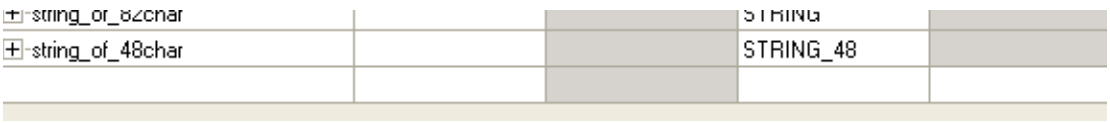
如果要创建非默认长度 （82 个字符）的字符串，请在项目树中右击“strings”（如下所示）。



然后，按如下所示配置属性。



完成后，即可定义新类型的实例。



对于类型 STRING 或 STRING_48 的实例，有一个 LEN 字段，在输入字符串常量或在 ASCII 或 STRING 指令操作字符串时，该字段会自动进行更新。

STEP 7 临时变量

STEP 7 中有一个变量类别是临时变量。在任何组织块、功能或功能块中都可以创建这种变量。

临时变量用于在本地临时存储中间值，也用于指针。这些变量仅在执行它们的块时存在，当块终止时，它们的值就会丢失。

Logix 没有临时变量。所有存储都是静态的，也就是说，在代码执行过程中会保留值。

如果使用附加指令，您会注意到可以为附加指令创建本地变量。这些变量的用法与临时变量相同。

功能

如果 STEP 7 程序员使用语句列表，则可能必须开发低级例程，编写这些例程很费时，并且需要仔细测试。功能非常重要，因为这类例程只需开发一次，在完成后，功能的开发者和其他程序员即可在很短的时间内实现该功能。

本节讨论如何在 Logix 中实现功能。

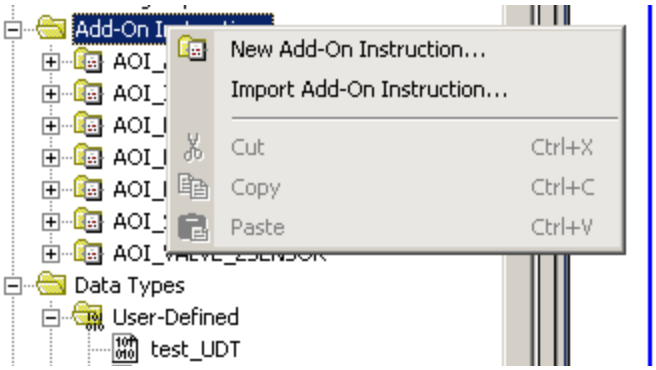
Logix 中作为附加指令的功能

STEP 7 功能和功能块在其结构方面与 Logix 附加指令类似。附加指令的参数类型 （Input、Output 和 InOut）与功能块相同，并且具有自己的数据区。经过编写和测试之后，附加指令即可在程序中的任何位置使用，因为独立程度很高，所以可以导出到其他项目或放置在代码库中。

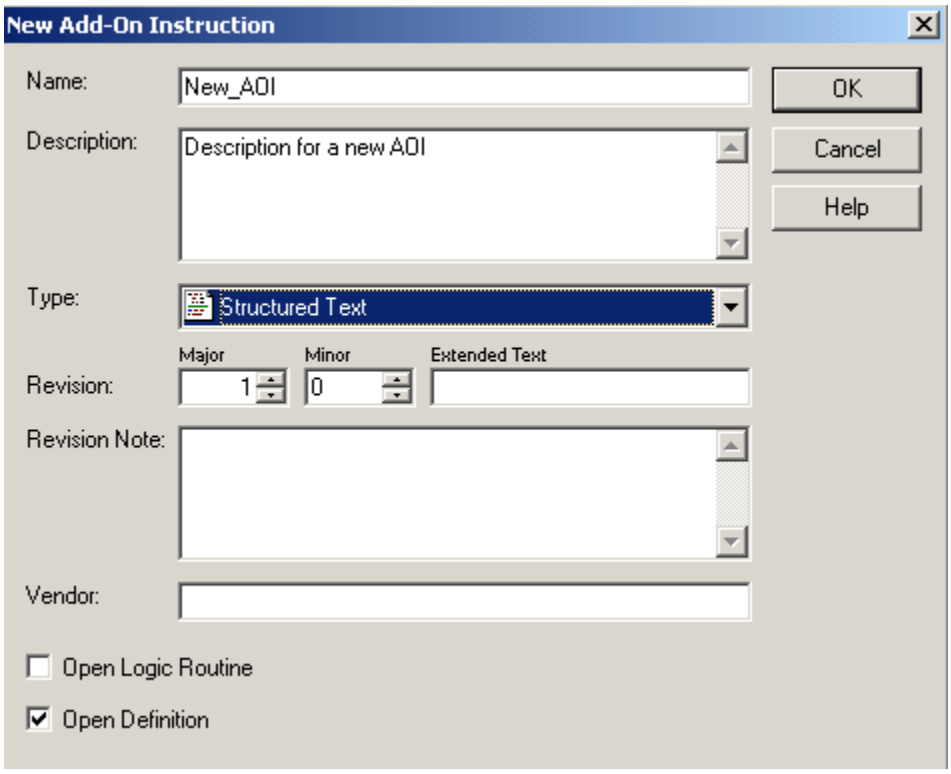
示例 - 斜坡功能

此示例使用一个实数变量，以指定斜率使该变量从当前值到达新值。

转到项目树的附加指令分支，右击附加指令。

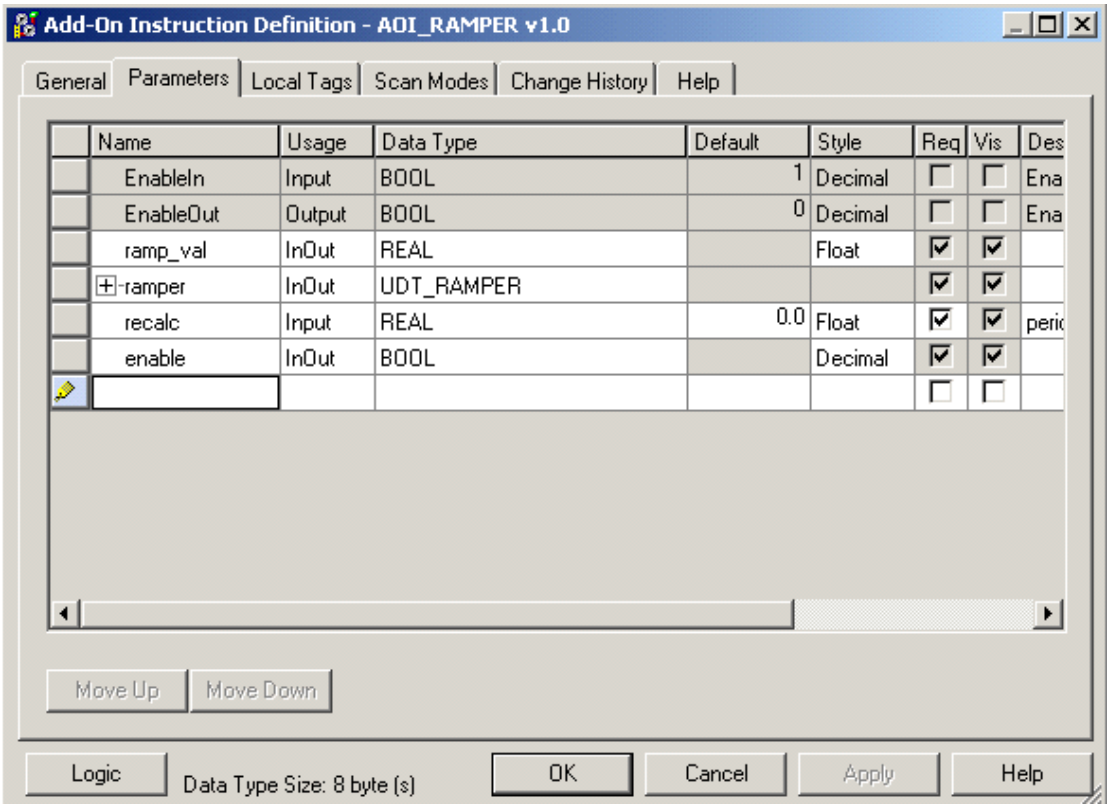


出现下面的窗体。

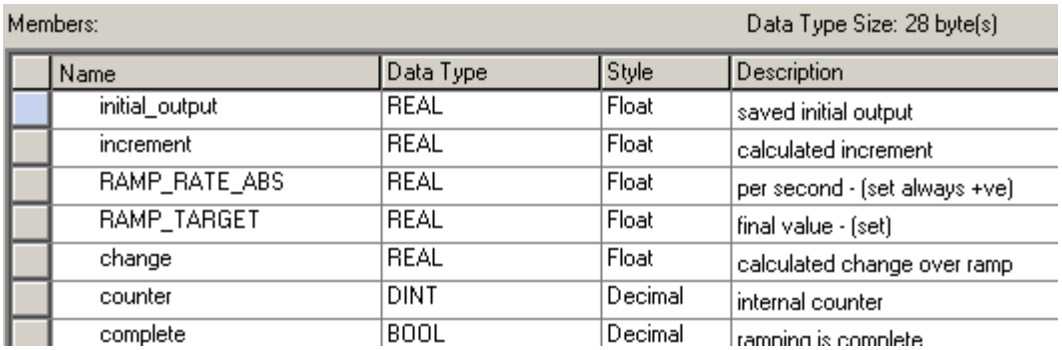


输入附加指令的名称，指定用于编写其代码段的语言。

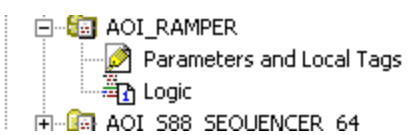
选择 Parameters 选项卡。



与在 STEP 7 中一样，Input 参数是从程序到附加指令的值，Output 参数是从附加指令到程序的值，InOut 参数用于附加指令将修改的变量。如果使用了任何数据结构，请选择 InOut 类型，因为数据结构是按引用传递的，这样会更为有效。



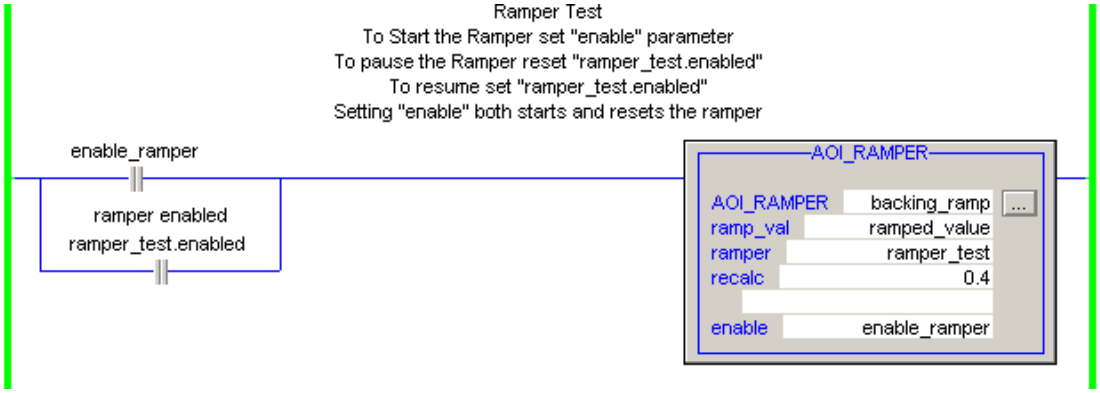
在 AOI_RAMPER 的项目树中，有一个逻辑部分。



打开它可查看此附加指令的代码。

```
// Ramps a real variable from its current valuse to a new value at a
// specified rate.

// Parameters:
//      ramp_val    - variable to be ramped
//      ramper      - instance of UDT UDT_RAMPER
//      recalc      - code recalculation period (s)
//      enable      - start signal

// To use - set the target value in ramper.RAMP_TARGET_ABS
//        - set the ramp rate in ramper.RAMP_RATE_ABS
//        - to Start the Ramper set "enable" parameter
//        - to pause the Ramper reset "ramper.enabled"
//        - to resume set "ramper.enabled"
//        - setting "enable" both starts and resets the ramper

// on completion, the UDT field "complete" is set and the UDT field
// "enabled" is reset

// when enable is set, initialise
if (enable & (enable xor ramper._enable)) then
    ramper.initial_output := ramp_val;
    ramper.change := ramper.RAMP_TARGET - ramp_val;
    ramper.increment := ramper.change / abs(ramper.change)
                              * ramper.RAMP_RATE_ABS * recalc;
    ramper.counter := 0;
    ramper.complete := 0;
    enable := 0;
    ramper.enabled := 1;
end_if;

ramper._enable := enable;

// ramp calculations
if (ramper.enabled) then
    ramp_val := ramper.initial_output + (ramper.counter
                                          * ramper.increment);
    ramper.counter := ramper.counter + 1;
    if  (abs(ramper.counter * ramper.increment)
                                       > abs(ramper.change)) then
         ramp_val := ramper.RAMP_TARGET;
         ramper.complete := 1;
         ramper.enabled := 0;
    end_if;
end_if;
```

附加指令可从任何例程调用。



注意，对于附加指令，需要在例程可见的数据区中创建附加指令类型的标签。这称为支持标签。

在编写附加指令之前，请查看 RSLogix 5000 软件中的指令帮助。可能会找到具有此功能的现有指令。下节将对此进行说明。

块复制、COP 和 CPS

在 STEP 7 中，通常使用系统功能 SFC20 “BLKMOV”在位置之间复制数据块。

```
CALL  "BLKMOV"
 SRCBLK :="Data_EM1".stepMsgs.step5
 RET_VAL:=#intVar
 DSTBLK :="Data_EM1".actualStep
```

该指令将字符串从字符串数组中的第五个位置复制到目标字符串。

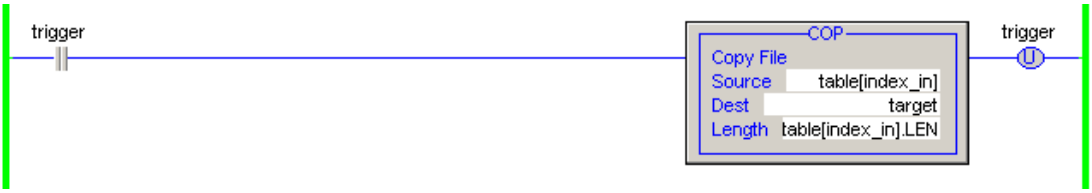
我们常常要从数组复制第 i 个元素，其中 “i”会随着程序的执行而变化。“BLKMOV”不能实现此功能。

STEP 7 程序员会编写一个功能来满足需要。

```
// copy step number descriptor to SCADA display area (EM faceplate)
      CALL  "INDEXED_COPY"
       indexSource :="Data_EMs".EM1.stepNumber
       sourceRef   :="Data_EM1".stepMsgs
       indexDest   :=1
       destRef     :="Data_EM1".actualStep
       recordLength:=8                  // 32 bytes
```

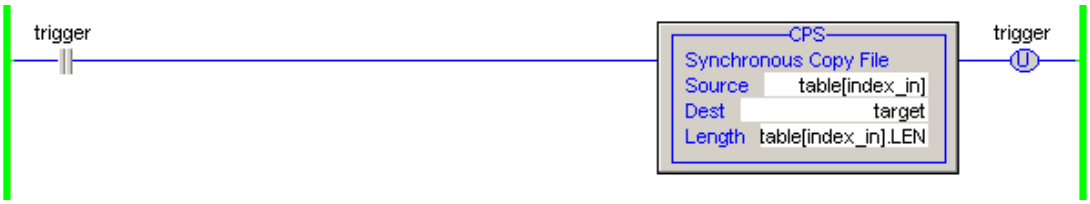
在这种情况下，复制在两个数组之间进行，索引由 indexSource 和 indexDest 定义。

在 Logix 中，COP 内置指令会保存所有工作。



因为指定的源和目标可以包含变量数组索引，所以 COP 将执行此功能。这等效于 “INDEXED_COPY”。

CPS 指令与 COP 相同，但有一点区别。



该指令无法中断。因此，在该指令执行过程中，源和目标数据保持不变。如果要移动可能更改的数据，请使用 CPS。

示例：

- 在程序要对数据进行操作的位置，将输入数据复制到缓冲区。
- 在程序要对数据进行操作的位置，将使用的标记复制到缓冲区。

数学表达式

本节介绍 S7 程序员如何执行 Logix 中的数学计算。将使用表达式 “v(cos(x)^2 + sin(x)^2)” 作为示例。此表达式的结果始终正好为 1，因此易于检查是否获得了正确的答案。

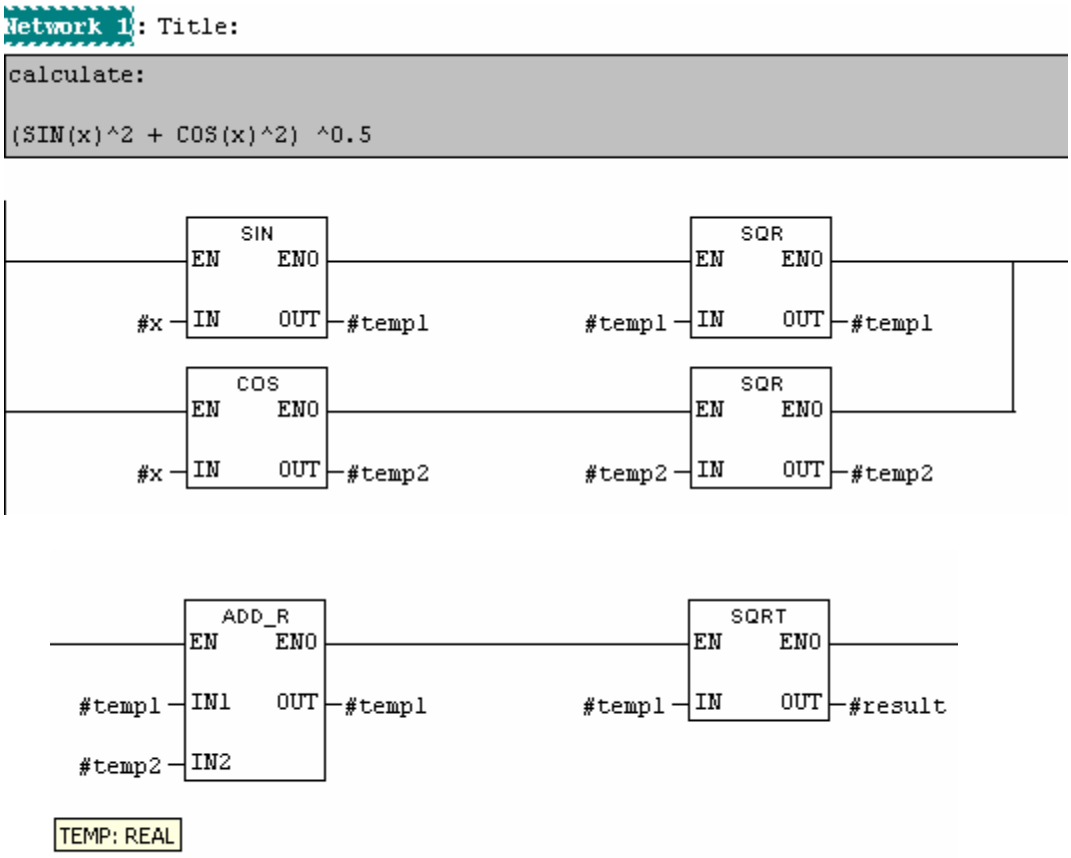
STEP 7 - STL

STEP 7 STL 中的数学代码十分高效，但对于不熟悉 STL 的人来说，可能不够清晰明了。

```
Network 1: Title:
calculate:

(SIN(x)^2 + COS(x)^2) ^0.5

          L     #x
          SIN
          SQR
          L     #x
          COS
          SQR
          +R
          SQRT
          T     #result
```

STEP 7 - LAD

LAD 中的数学计算遵循传统的功能组合模式。

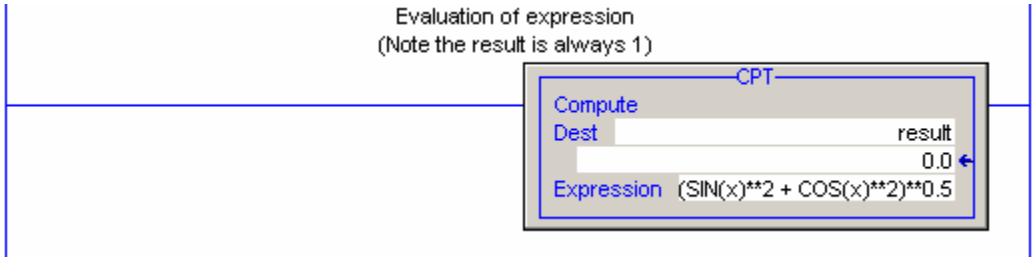


Logix - ST

表达式的输入方式与其他任何高级语言相同。

```
// evaluation of mathematical expression in Structured Text

result := (SIN(x)**2 + COS(x)**2)**0.5;
```

Logix - LD



使用 CPT 指令可以以高级方式输入表达式，与不同指令网络 （梯级）相比，大多数人更容易理解这种方式。

STEP 7 - 用户功能

编写此功能块是为了完成与 Logix CPT 十分相似的功能。

```
CALL  "CPT" , "Data_CPT"                    FB119 / DB119
 str   :="test_formulae".test_string15      P#DB16.DBX476.0
 x     :=3.300000e+001
 a     :=0.000000e+000
 b     :=0.000000e+000
 c     :=0.000000e+000
 result:=#realVar
```

```
-- ((cos x)^2 + (sin x)^2) ^0.5 --
```

IN	OUT
+3.300e+001 +0.000e+000 +0.000e+000 +0.000e+000	+1.000e+000

此功能块读取并计算存储在数据块中的表达式字符串。与 Logix CPT 相比，它有一个限制，即表达式以逆波兰表示法编写，不是所有人都适应这种方式。

编写这类功能块的主要问题是要花费不少时间，不适于初级程序员。对于 Logix，CPT 指令适合所有人使用，只要安装了 RSLogix 5000 软件即可。

类型检查

STEP 7 和 Logix 的编译程序都会对功能、功能块、指令和附加指令的参数进行严格的类型检查。

在数学表达式方面，则存在一些差异。

Logix 区分 Numeric 和 Boolean 值。编译程序会拒绝不合逻辑地混合了数字和布尔值的表达式。当编译程序遇到混合数字类型的表达式时，会进行转换以生成类型为已声明结果变量的结果。因此，如果结果应为整数，则将 * 解释为整数乘法，如果结果应为实数，则解释为实数乘法。

在 STEP 7 中，必须指定算术操作的类型。例如 *I （两个 16 位整数相乘）、*D （两个 32 位整数相乘）和 *R （两个实数相乘）。由程序员确保作为 *R 指令的操作数的两个数字为实数。如果不是实数，则编译程序不会提示，但结果是没有意义的。

结论

Logix 的数学表达式编程方法更为清晰，而且通过使数学代码与其他逻辑分离，可简化测试和验证。

与编程有关的其他主题

变量的作用域

在这一点上，Logix 与 STEP 7 有显著差异。

STEP 7 的规则

- 在声明临时变量的块之外，临时变量是不可见的。
- 全局静态变量在整个程序中都可见。
- 声明为功能块的实例数据的静态变量在功能块中具有特殊状态，但从程序的其他部分可以对它们进行存取。

Logix 的规则

Logix 中的执行划分为多个任务。每个任务可以有多个程序，每个程序又可以有多个例程。每个程序可以有自己的标签段。

- 控制器作用域标签在所有程序的所有例程中都可见。
- 程序作用域标签仅在定义这些标签的程序中的例程中可见。这意味着如果一个程序中的某个例程与另一个程序中的例程共享数据，则该例程必须使用控制器作用域数据。
- 附加指令本地标签仅对附加指令的逻辑可见。

OB、任务和调度

第 2 章介绍了组织块、任务和调度。

一个更大的示例 - 控制模块

此示例将前面各节中说明的部分主题结合起来。术语 “控制模块” (CM) 源自有重要影响的 S88 批处理控制标准。S88 鼓励控制器软件设计更加 “面向对象”。此控制模块用于二进制值。附加指令适用于这类编程。

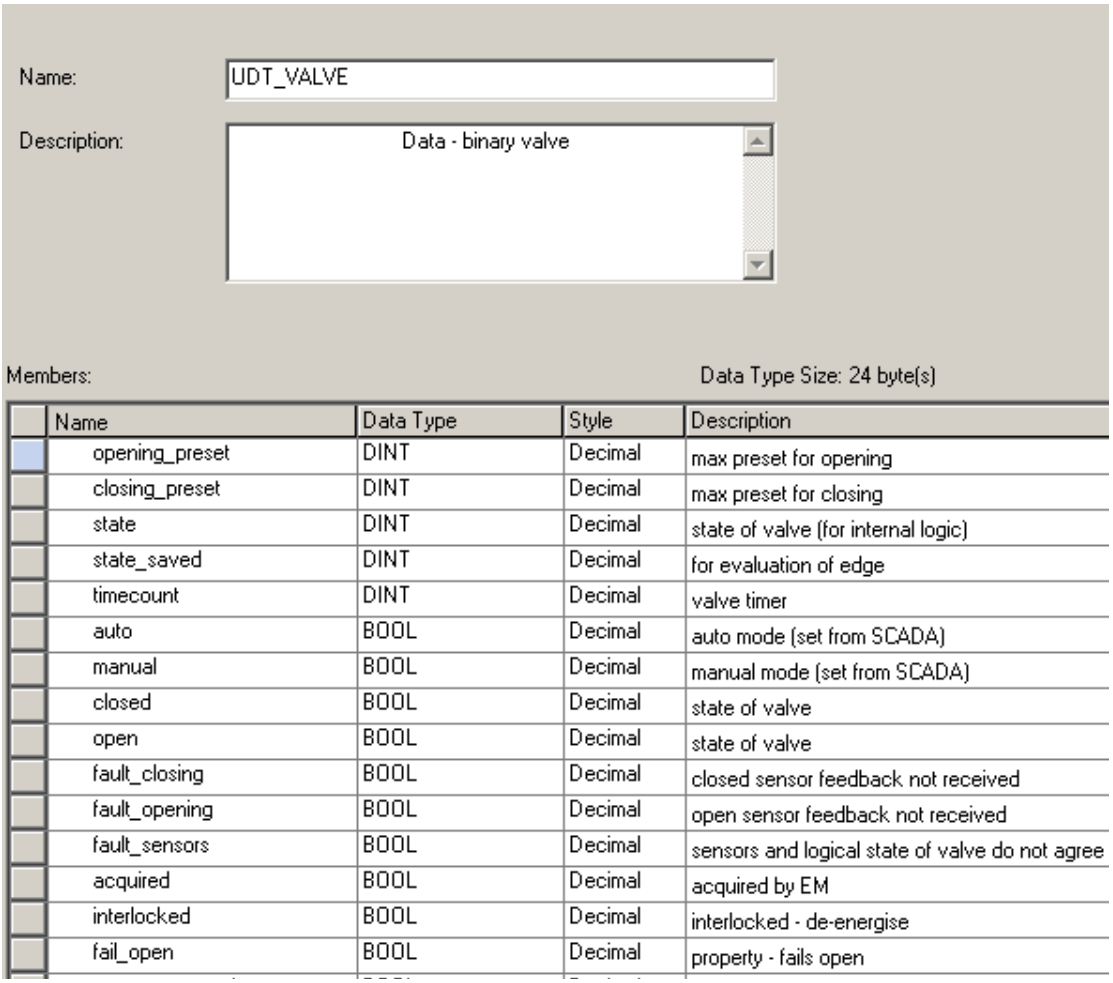
CM 的组件

这些组件是：

- 名为 UDT_VALVE 的 UDT。
- 名为 AOI_VALVE_2SENSOR 的附加指令。
- 位于 “task_02s” 之下、名为 “valves_callup” 的新程序，该程序包含程序标签段和一个例程。

用户数据类型阀

该 UDT 如下所示。



Name: UDT_VALVE

Description: Data - binary valve

Members:　　Data Type Size: 24 byte(s)

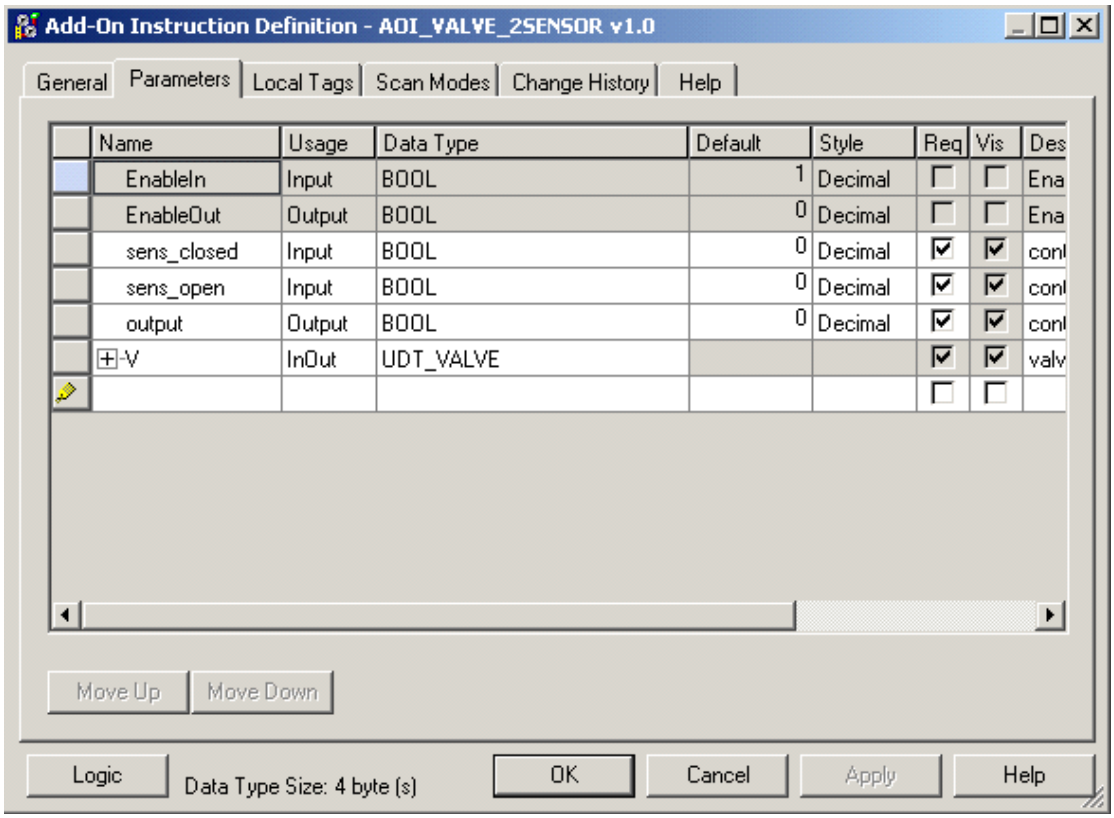
Name	Data Type	Style	Description
opening_preset	DINT	Decimal	max preset for opening
closing_preset	DINT	Decimal	max preset for closing
state	DINT	Decimal	state of valve (for internal logic)
state_saved	DINT	Decimal	for evaluation of edge
timecount	DINT	Decimal	valve timer
auto	BOOL	Decimal	auto mode (set from SCADA)
manual	BOOL	Decimal	manual mode (set from SCADA)
closed	BOOL	Decimal	state of valve
open	BOOL	Decimal	state of valve
fault_closing	BOOL	Decimal	closed sensor feedback not received
fault_opening	BOOL	Decimal	open sensor feedback not received
fault_sensors	BOOL	Decimal	sensors and logical state of valve do not agree
acquired	BOOL	Decimal	acquired by EM
interlocked	BOOL	Decimal	interlocked - de-energise
fail_open	BOOL	Decimal	property - fails open

首先应生成该 UDT，它包含对阀进行建模所需的所有数据。

附加指令

附加指令参数

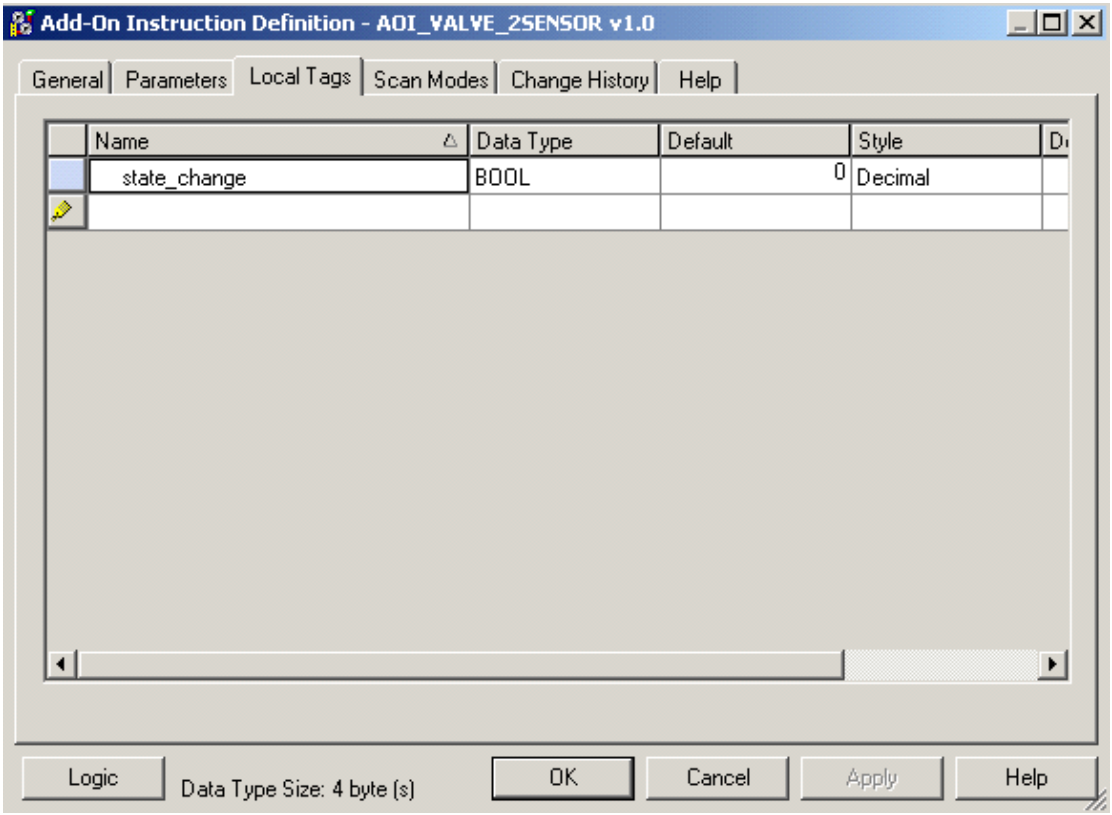
屏幕截图显示参数配置屏幕。



已添加的参数是阀的 I/O 和一个 “UDT_VALVE”类型的对象。“V”必须是 InOut 参数。

附加指令本地数据

下面的屏幕截图显示附加指令本地数据的配置页。



附加指令逻辑

下面的屏幕截图显示此附加指令的逻辑。

```
// Control Module Valve_2sensor
// ----------------------------
// Implements logic for a valve with an open and a closed sensor and one output

// See UDT Valve for data structure.

// Note the open/close command V.open_command must be set or reset externally
// and then left until the next activation is required. Do not continuously
// hold the flag set or reset.

// increment timer counter
V.timecount := V.timecount + 1;

// evaluate change of state (state machine)
state_change := V.state <> V.state_saved;
V.state_saved := V.state;

// set output
output := (V.fail_open xor V.open_command) and not
                                        (V.interlocked or V.faulted);

// valve is faulted
V.faulted := V.fault_opening or V.fault_closing or V.fault_sensors;

// action on fault or interlock
if V.faulted or V.interlocked then
    if V.fail_open then
        V.state := 3;
        V.open_command := 1;
    else
        V.state := 0;
        V.open_command := 0;
    end_if;
end_if;


// state machine:
// the state machine does not set outputs - it monitors inputs
// to set status and faults.
case V.state of
    // state 0 - valve is closed - wait for open command
    0:  V.closed := 1;
        V.open := 0;
        if (V.open_Command) then
            V.state := 1;
        // fault sensors
        else
            V.fault_sensors := (not sens_closed) or (sens_open);
        end_if;
    // state 1 -
    1:  V.state := 2;
    // state 2 - waiting for open sensor
    2:  if (sens_open & not sens_closed) then
            V.state := 3;
```

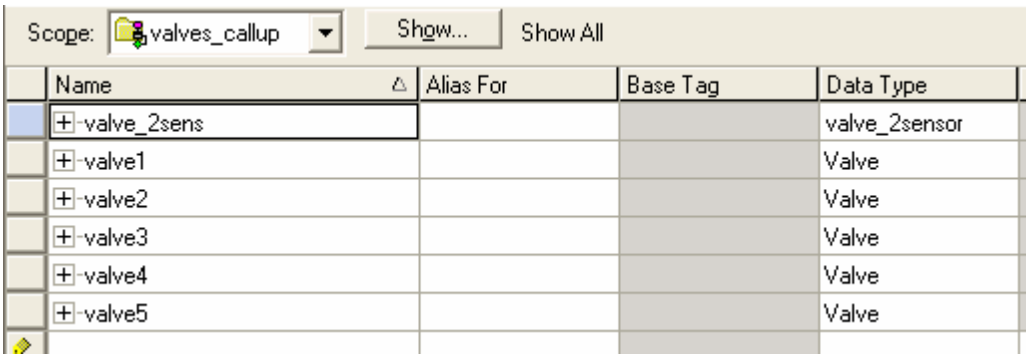
```
        // possible close command while waiting to open
        elsif not V.open_command then
            V.state := 0;
        // fault opening
        else
            V.fault_opening := (V.timecount >  V.opening_preset);
        end_if;
    // state 3 - open - wait for close command
    3:  V.closed := 0;
        V.open := 1;
        if (not V.open_command) then
            V.state := 4;
        // fault sensors
        else
            V.fault_sensors := (sens_closed) or (not sens_open);
        end_if;
    // state 4 -
    4:  V.state := 5;
    // state 5 - wait for closed sensor
    5:  if (sens_closed & not sens_open) then
            V.state := 0;
        // possible open command while waiting to close
        elsif V.open_command then
            V.state := 3;
        // fault closing
        else
            V.fault_closing := (V.timecount >  V.closing_preset);
        end_if;

          _
else;
end_case;
// end state machine

// reset timer if change of state
if (state_change) then V.timecount := 0;
end_if;

// external fault reset
if (V.clear_faults) then
    V.fault_opening := 0;
    V.fault_closing := 0;
    V.fault_sensors := 0;
    V.clear_faults := 0;
end_if;
```

此逻辑中引用的标签是所有参数或本地标签。这意味着可以在任何程序中使用该附加指令（如果 UDT 阀也存在）。

调用

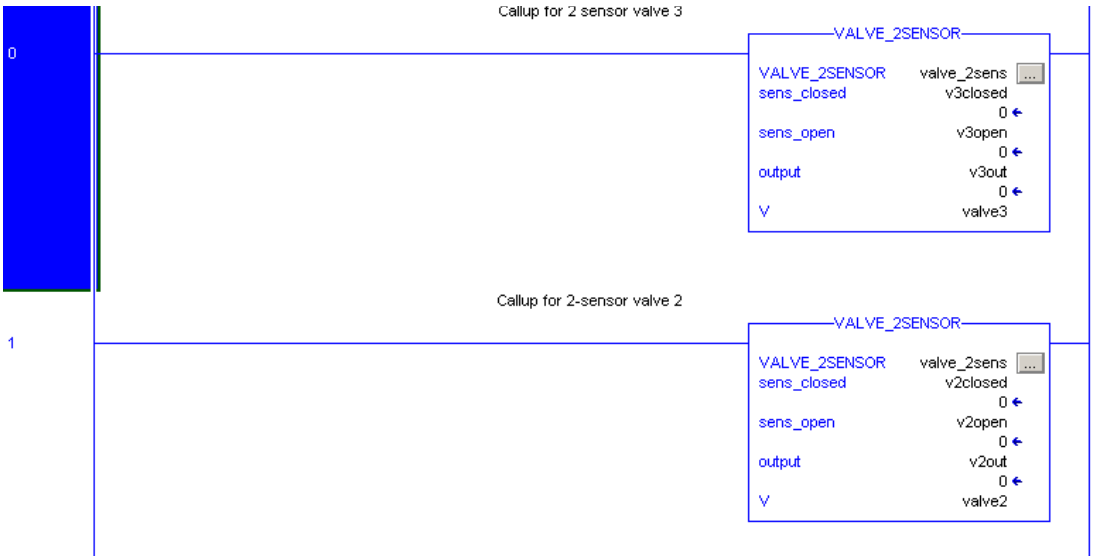
调用代码和 UDT 阀的实例都位于程序 “valves_callup”中，该程序在 task_02s 之下运行。执行调用代码的频率取决于应用和阀的大小。

下面的屏幕截图显示数据实例。



为每个物理阀都添加一个 Valve 类型的实例。第一个标签是附加指令必需的 “支持标签”。

下面的屏幕截图显示调用代码。



对每个阀调用一次附加指令。实际参数是阀的传感器和螺线管的实际 I/O 标签，以及 UDT “阀”的实例。

I/O 标签仅在对附加指令的调用中出现。程序中的任何其他位置都不会使用这些标签。除了从软件结构的角度看更整洁之外，还可避免由 I/O 的异步更新所引起的问题。

请记住，对于 Logix 控制器，I/O 扫描是异步进行的。

注：

转换到 Logix 时的常见错误

简介

本节的目的是指出 S7 用户在将应用程序转换到 Logix 时常犯的一些设计和编程错误。这些错误是通过对从 STEP 7 转换而来的 Logix 程序进行检查所确定的。

编程错误分为两类：

- 会导致降低控制器效率的编程。
- 会导致难以理解、维护和开发的控制系统的编程。

在大多数情况下，高效的编码也会提高程序的可读性和模块化。反过来说，改善程序结构也会提高程序效率。

未选择合适的硬件

本章主要关注软件。不过请记住，正确选择硬件是获得满意操作的前提。控制器和机架的数量可能与等效的 S7 系统不同。

有关硬件的更多信息，请阅读第 1 章和附录 A。有关更多信息，请参见附录 A 和 B。

低估任务调度的影响

在调度和中断方面，这两种系统的功能没有太大差异。但在 Logix 领域中，更积极鼓励进行调度。

STEP 7 程序员在使用 Logix 控制器时常常会忽视调度。有关 Logix 调度的详尽信息，请参见第 2 章。

进行转译而不是转换

将 STEP 7 程序逐行转译为 Logix 是一种常见错误。

相反，需要的是更加细致的过程，这一过程称为 “转换”。这涉及语言的选择、调度和代码例程的选择。

通过转换而不是转译 STEP 7 程序，可以更好地利用 Logix 系统的性能。

未使用最适当的 Logix 语言

程序员通常关注梯形图逻辑，而忽视了 Logix 语言。

有关如何选择 Logix 语言的讨论，请阅读第 2 章，有关 STEP 7 代码转换为 Logix 的示例，请阅读第 4 章。

实现不正确的数据类型 - DINT 与 INT

通常建议使用 DINT 而不是 INT。

下面的示例说明两个 DINT 相加与两个 INT 相加。

DINT 相加

```
// add two DINTs and assign to a third DINT

for index := 0 to 999 do
    result_DINT := operandA_DINT + operandB_DINT;
end_for;
```

INT 相加

```
// add two INTs and assign to a third INT

for index := 0 to 999 do
    result_INT := operandA_INT + operandB_INT;
end_for;
```

计时结果

下表列出了相对时间 （数字越小表示速度越快）。此处的数字仅用于与表中的其他数字进行比较。这些数字不应与其他表中的条目进行比较。

方法	相对时间
通过 ST For 循环进行 DINT 加法	53
通过 ST For 循环进行 INT 加法	100

为进行比较，对 S7 控制器进行了同样的测试。在本例中，DINT 和 INT 的结果是相同的。

结论是在 Logix 中所有整数都应使用 DINT。仅当与要求使用 INT 或 SINT 的外部系统连接时才使用 INT 或 SINT。

模拟现有指令的用户代码

程序员经常编写用户代码去完成现有指令可以完成的功能。以复制数组为例，将用户代码与 COP 指令进行比较。

用户代码

```
for index := 0 to 99 do
    target_DINT[index] := source_DINT[index];
end_for;
```

COP 指令

```
cop(source_DINT[0], target_DINT[0], 100);
```

下面是两种方法的相对计时。此处的数字同样仅用于与表中的其他数字进行比较。这些数字不应与其他表中的条目进行比较。

方法	相对计时
使用结构文本复制 DINT 数组	100
使用 COP 复制 DINT 数组	18

若要执行复制数组这样的操作，可使用语句列表中编写的 STEP 7 库函数。如果库函数无法完成所需功能，则可以编写新的函数。编写的函数几乎与 STEP 7 提供的函数一样高效。

但在 Logix 中，程序员不可能编写出与内置 COP 一样高效的复制函数。因此，S7 程序员在编写函数之前，应仔细查看 RSLogix 5000 软件中的指令帮助。

未正确使用 COP、MOV 和 CPS

MOV 将简单值 （立即数或标签）复制到简单标签类型 — DINT、INT、SINT 或 REAL。COP 可以完成与 MOV 一样的功能 （源不能是立即数值），但它更重要的用途是复制复杂数据类型。

使用 COP 复制简单数据类型是编程中的小错误。

一种常见错误是在可以使用一个 COP 时使用多个 MOV 复制数据结构。

如果因异步 I/O 更新而使源数据在复制过程中可能发生更改，请改用 CPS。此指令无法中断，因此在复制过程中源数据保持不变。

未正确使用 CPT

在 Logix 中，CPT 指令可以用于评估表达式。表达式是在该指令的一个字段中输入的。这十分方便。

但是，仅当需要使用多条算术指令来评估表达式时，才应使用 CPT。如果单条指令够用，则该指令的速度比 CPT 快。

有关 CPT 的更多信息，请参见第 4 章。

未以最佳方式处理字符串

如果要定义新的 String 类型 （例如字符数不是默认值 82 的类型），创建新的 “用户数据类型” 则是错误的。相反，应创建新的 String 数据类型。采用这种方法的优点在于，“LEN” 字段会随字符串长度的变化自动更新。

大量使用转移

在 Logix 中，只有在梯形图逻辑中才能使用转移。建议尽量不用 JMP 指令。梯形图逻辑中的转移常常使程序难以理解。

未使用别名标签

请记住为 RSLogix 5000 软件所创建的 I/O 标签创建别名标签。这些标签可使程序更易于理解。请参见第 2 章。

S7 与 Logix 术语对照表

简介

本章介绍 S7 术语及对应的 Logix 术语。

硬件术语

S7 术语	定义	Logix 术语	定义
通讯处理器	通讯模块	桥接器	
控制器	控制器	控制器	
CPU	中央处理单元	CPU 或控制器	
防故障 CPU	CPU 315F-2 DP 实现 PROFISAFE 版本的 DP	GuardLogix	L61S、L62S、L63S
工业以太网	Siemens 版本的以太网	EtherNet/IP ControlNet	两者都具有与工业以太网相同（或更好）的功能
MPI	多点接口 ？一种串行总线	串行	DF1 或 DH485 协议
可编程控制器		控制器或 PAC	
PROFIBUS DP	常用现场总线	EtherNet/IP ControlNet DeviceNet	
PROFIBUS PA	各种专用于过程自动化的 Profibus	作为 Profibus DP	
PROFINET	以太网上的 Profibus	EtherNet/IP	
PROFISAFE	防故障型 PROFIBUS DP	GuardLogix	
S7-200	低端控制器	MicroLogix	
S7-300	中级控制器	CompactLogix	
S7-400	高端控制器	ControlLogix	
SIMATIC	Siemens 自动化产品的品牌名称	Logix	

软件术语

S7 术语	定义	对应的 Logix 术语	定义
累加器	在 STL 中使用	无	在 Logix 语言中，无需访问 CPU 的低级结构
AR1、AR2	指针寄存器	无	在 Logix 语言中，无需访问 CPU 的低级结构
数组	语法 ARRAY[0…7] OF REAL	数组	语法 REAL[8] 始终从 0 开始进行索引
位内存	地址 M...	无	使用标签
块传输	复制数据块。 SFC20 BLK_MOV	COP	指令 （对简单变量使用 MOV）
BOOL		BOOL	
BYTE	8 位字	SINT	除非有必要 （如字符串的字符），否则不建议使用 （因为速度比 DINT 慢）
CFC	可选过程控制语言	FBD	标准功能块语言。
CHAR	作为字符的字节	SINT	
Cycle_Execution	OB1 - 连续执行	连续任务	连续执行
数据块	静态数据内存单元	控制器作用域标签数据库 或程序作用域标签数据库	全局 在数据库链接到的程序中可见
DINT	双精度整数	DINT	双精度整数
DWORD	32 位字	DINT	
FBD	功能块图	FBD	功能块图
功能	具有临时内存但是没有静态内存的程序单元	例程 AOI （附加指令）	两者都可能对应于功能
功能块	具有临时内存和静态内存的程序单元	例程 AOI （附加指令） 程序	这些都**可能**对应于功能块
GRAPH	可选图形语言	顺序功能图	标准图形语言
HW 配置	硬件配置 - STEP 7 的组件	I/O 配置	控制器组织器的分支
INT	整数	INT	不建议使用 （它的速度比 DINT 慢）
Interrupt_Execution	周期性执行组织块	周期性任务	周期性执行的任务
LAD	梯形图逻辑	LD	梯形图逻辑
库	系统功能	GSV、SSV	指令 获取系统值 设置系统值
NetPro	网络配置器	无	控制器组织器的 I/O 配置分支的组成部分。

S7 术语	定义	对应的 Logix 术语	定义
组织块	由操作系统调用的程序单元	任务	由操作系统调用的程序单元
指针	在 STL 中使用的数据指针	无	使用数组
REAL	32 位浮点数	REAL	32 位浮点数
SCL	可选高级语言	结构文本	标准语言
Simatic 管理器	STEP 7 的组件	控制器组织器	RSLogix 5000 的组件
STEP 7	S7 的开发和监控软件	RSLogix 5000	Logix 的开发和监控软件
STL	语句列表	无	使用结构文本、梯形图逻辑或顺序功能图
STRING	CHAR 的序列。默认长度为 254	STRING	SINT 的序列。默认长度为 82。String 对象还包含其长度作为属性 .LEN
STRUCT	数据的非类型化集合	无	在 Logix 中，结构是类型 (UDT) 的实例
符号	数据内存地址的名称	标签	标签定义变量的结构*并* 保留内存
临时内存	在运行时堆栈上创建的内存	无	使用标签
WORD	16 位字	INT	
UDT	用户数据类型	UDT	用户数据类型

注：

S7 300 和 S7 400 部件与罗克韦尔自动化同等产品

简介

本附录列出了 Siemens 产品及其罗克韦尔自动化的同等产品。

紧凑型 S7 300 CPU

Siemens 产品目录号	Siemens 简短参考号	内存		通讯端口		最大 MMC 大小		嵌入式 I/O			RA 解决方案
			MPI	DP	串行		DI	DO	AI	AO	
6ES7 312-5BE0x-xxxx	S7-312C	32K	有	无	无	4 MB	10	6			1769-L31 + Compact I/O ML1500
6ES7 313-5BF0x-xxxx	S7-313C	64K 有 无 无	有	无	无	8 MB	24	16	4	2	1769-L31 + Compact I/O ML1500
6ES7 313-6BF0x-xxxx	S7-313C-PtP	64K	有	无	RS422/485	8 MB	16	16			1769-L31 + Compact I/O ML1500
6ES7 313-6CF0x-xxxx	S7-313C-DP	64K	有	有	无	8 MB	16	16			1769-L31 + Compact I/O ML1500
6ES7 314-6BG0x-xxxx	S7-314C-PtP	96K	有	无	RS422/485	8 MB	24	16	4	2	1769-L31 + Compact I/O ML1500
6ES7 314-6CG0x-xxxx	S7-314C-DP	96K 有 无 8 MB	有	有	无	8 MB	24	16	4	2	1769-L31 + Compact I/O ML1500

标准 S7 300 CPU

Siemens 产品目录号	Siemens 简短参考号	内存		通讯端口		最大负载内存大小 (RAM)	RA 解决方案
			MPI	DP	PN		
6ES7 312-1AE1x-xxxx	S7-312	32K	有	无	无	4 MB	1769-L31
6ES7 314-1AG1x-xxxx	S7-314	96K	有	无	无	8 MB	1769-L31
6ES7 315-2AG1x-xxxx	S7-315-2 DP	128K	有	有	无	8 MB	1769-L3xE 或 1769-L3xC
6ES7 315-2EH1x-xxxx	S7-315-2 PN/DP	256K	有	有	有	8 MB	1769-L3xE 或 1769-L3xC

Siemens 产品目录号	Siemens 简短参考号	内存		通讯端口		最大负载内存大小 (RAM)	RA 解决方案
			MPI	DP	PN		
6ES7 317-2AJ1x-xxxx	S7-317-2 DP	512K	有	有	无	8 MB	1769-L3xE 或 1769-L3xC
6ES7 317-2EK1x-xxxx	S7-317-2 PN/DP	1 MB	有	有	有	8 MB	1769-L3xE 或 1769-L3xC
6ES7 319-3ELOx-xxxx	S7-319-3 PN/DP	1.4 MB	有	有	有	8 MB	1769-L3xE 或 1769-L3xC

技术型 S7 300 CPU

Siemens 产品目录号	Siemens 简短参考号	内存		通讯端口		最大负载内存大小 (RAM)	RA 解决方案
			MPI	DP	PN		
6ES7 315-6TG1x-xxxx	S7-315T-2 DP	128K	有	有	有	4 或 8 MB	1768-L43
6ES7 317-6TJ1x-xxxx	S7-317T-2 DP	512K	有	有	有	4 或 8 MB	1768-L43

防故障型 S7 300 CPU

Siemens 产品目录号	Siemens 简短参考号	内存		通讯端口		最大负载内存大小 (RAM)	RA 解决方案 ControlLogix
			MPI	DP	PN		
6ES7 315-6FF1x-xxxx	S7-315F-2 DP	192K	有	有	无	8 MB	GuardLogix 或 SmartGuard 600
6ES7 315-2FH1x-xxxx	S7-315F-2 PN/DP	256K	有	有	有	8 MB	GuardLogix 或 SmartGuard 600
6ES7 317-6FFOx-xxxx	S7-317F-2 DP	1 MB	有	有	无	8 MB	GuardLogix 或 SmartGuard 600
6ES7 317-2FK1x-xxxx	S7-317F-2 PN/DP	1 MB	有	有	有	8 MB	GuardLogix 或 SmartGuard 600

S7 300 数字输入模块

Siemens 产品目录号	前端连接器	点数	范围	RA 解决方案	备注
6ES7 321-1BH0x-xxxx	20 针	16	24 VDC	1769-IQ16 1769-IQ16F	
6ES7 321-1BH5x-xxxx	20 针	16	24 VDC	1769-IQ16 1769-IQ16F	
6ES7 321-1BL0x-xxxx	40 针	32	24 VDC	1769-IQ32 1769-IQ32T	
6ES7 321-1CH0x-xxxx	40 针	16	24 … 48 V	无	
6ES7 321-1CH2x-xxxx	20 针	16	48 … 125 VDC	无	
6ES7 321-1BH1x-xxxx	20 针	16	24 VDC	1769-IQ16 1769-IQ16F	
6ES7 321-7BH0x-xxxx	20 针	16	24 VDC	1769-IQ16 1769-IQ16F	
6ES7 321-1FH0x-xxxx	20 针	16	120 … 230 VAC	1769-IA16	1769-IA16 只接受 120 VAC
6ES7 321-1FF0x-xxxx	20 针	8	120 ... 230 VAC	1769-IM12	1769-IM12 只接受 230 VAC
6ES7 321-1FF1x-xxxx	40 针	8	120 ... 230 VAC	1769-IA8I	1769-IA8I 只接受 120 VAC
6ES7 321-1EL0x-xxxx	40 针	32	120 VAC	无	
无		16	5 VDC TTL	1769-IG16	

S7 300 数字输出模块

Siemens 产品目录号	前端连接器	点数	范围	输出电流	RA 解决方案	备注
6ES7 332-1FH0x-xxxx	20 针	16	120/230 VAC	0.5 A	1769-OA16	
6ES7 332-1FF0x-xxxx	20 针	8	120/230 VAC	2 A	1769-OA8	S7-300 每个组都有熔丝
6ES7 332-5FF0x-xxxx	40 针	8	120/230 VAC	2 A	1769-OA8	S7-300 出现在组 1 中
6ES7 322-1BH0x-xxxx	20 针	16	24 VDC	0.5 A	1769-OB16 1769-OB16P	
6ES7 322-1BH1x-xxxx	20 针	16	24 VDC	0.5 A	无	高速
6ES7 322-1BL0x-xxxx	40 针	32	24 VDC	0.5 A	1769-OB32 1769-OB32T	
6ES7 322-1BF0x-xxxx	20 针	8	24 VDC	2 A	1769-OB8	
6ES7 322-8BF0x-xxxx	20 针	8	24 VDC	0.5 A	1769-OB8	
6ES7 332-1FL0x-xxxx	2x20 针	32	120 VAC	1 A	无	
6ES7 332-5GH0x-xxxx	40 针	16	24/48 V	0.5 A	无	
6ES7 332-1CF0x-xxxx	20 针	8	48 … 125 VDC		无	
无		16	5 VDC TTL		1769-OG16	

无		16	24 VDC		1769-OV16	
无		32	24 VDC		1769-OV32T	
无		16	24 VDC		1769-OB16P	

S7 300 继电器输出模块

Siemens 产品目录号	前端连接器	点数	输出电流	RA 解决方案	备注
6ES7 322-1HH0x-xxxx	20 针	16	2 A	1769-OW16	
6ES7 322-1HF0x-xxxx	20 针	8	5 A	1769-OW8	
6ES7 322-1HF1x-xxxx	40 针	8	5 A	1769-OW8I	
6ES7 322-5HF0x-xxxx	40 针	8	8 A	1769-OW8I	S7-300 模块附带 RC 滤波器和过电压保护

S7 300 数字组合模块

Siemens 产品目录号	前端连接器	点数	范围输入	输出电流	RA 解决方案	备注
6ES7 323-1BH0x-xxxx	20 针	8 / 8	24 VDC	24 VDC / 0.5 A	1769-IQ6XOW4	Compact I/O 的 I/O 更少，输出为继电器
6ES7 323-1BL0x-xxxx	40 针	16 / 16	24 VDC	24 VDC / 0.5 A	无	
6ES7 327-1BH0x-xxxx	20 针	8 / 8	24 VDC	24 VDC / 0.5 A	无	8 路输入；8 路输入或输出（可配置）

S7 300 模拟输入模块

Siemens 产品目录号	**前端连接器**	**点数**	**分辨率 （位）**	**类型**	**Compact I/O 解决方案**	**备注**
6ES7 331-1KF0x-xxxx	40	8	13	电压，电流，电阻 温度	1769sc-IF8U 1769-IF8U	
6ES7 331-7KF0x-xxxx	20	8	9 / 12 / 14	电压，电流，电阻 温度	1769sc-IF8U 1769-IF8U	
6ES7 331-7KB0x-xxxx	20	2	9 / 12 / 14	电压，电流，电阻 温度	1769sc-IF8U 1769-IF4	
6ES7 331-7NF0x-xxxx	40	8	16	电压 电压	1769-IF8	

6ES7 331-7NF1x-xxxx	40	8	16	电压 电压	1769-IF8	包括周期结束时的硬件中断和 6ES7 331-7NF0x-xxxx
6ES7 331-7HF0x-xxxx	20	8	14	电压 电压	1769-IF8	
6ES7 331-7PF0x-xxxx	40	8		RTD 电阻	1769-IR6	
6ES7 331-7PF1x-xxxx	40	8		热电偶	1769-IT6	
无					1769-IF4I	

S7 300 模拟输出模块

Siemens 产品目录号	前端连接器	点数	分辨率 （位）	类型	RA 解决方案	备注
6ES7 332-5HD0x-xxxx	40	4	12	电压 电流	1769-OF4VI 1769-OF4CI	
6ES7 332-7ND0x-xxxx	20	4	16	电压 电流	1769-OF4VI 1769-OF4CI	
6ES7 332-5HB0x-xxxx	20	2	12	电压 电流	1769-OF2	
6ES7 332-5HF0x-xxxx	20	8	12	电压 电流	1769-OF8V 1769-OF8C	

S7 300 模拟组合模块

Siemens 产品目录号	前端连接器	点数	分辨率 （位）	类型	RA 解决方案	备注
6ES7 334-0KE0x-xxxx	20	4 / 2	12	电压 电流 Pt 100		只输出电压
6ES7 334-0CE0x-xxxx	20	4 / 2	8	电压和电流 （输入和输出）	1769-IF4XOF2	

S7 400 标准控制器

Siemens 产品目录号	Siemens 简短参考号	工作内存大小		通讯端口		最大负载内存大小 (RAM)	RA 解决方案 ControlLogix
			MPI	DP	PN		
6ES7 412-1XF04-0AB0	CPU 412-1	144KB	有	有	无	64MB	1756-L61
6ES7 412-2GX04-0AB0	CPU 412-2	256KB	有	有	无	64MB	1756-L61
6ES7 414-2GX04-0AB0	CPU 414-2	512KB	有	有	无	64MB	1756-L62
6ES7 414-3XJ04-0AB0	CPU 414-3	1.4 MB	有	有	无	64MB	1756-L63
6ES7 414-3EM05-0AB0	CPU 414-3 PN/DP	2.8MB	有	有	有	64MB	1756-L63
6ES7 416-3XK04-0AB0	CPU 416-2	2.8MB	有	有	无	64MB	1756-L63
6ES7 416-3XL04-0AB0	CPU 416-3	5.6MB	有	有	无	64MB	1756-L64
6ES7 416-3ER05-0AB0	CPU 416-3 PN/DP	11.2 MB	有	有	有	64MB	1756-L64
6ES7 417-4XL04-0AB0	CPU 417-4	20MB	有	有	无	64MB	1756-L64

冗余和防故障控制器

Siemens 产品目录号	Siemens 简短参考号	工作内存大小		通讯端口			最大负载内存大小 (RAM)	RA 解决方案 ControlLogix
			MPI	DP	PN	同步端口		
6ES7 414-4HJ04-0AB0	CPU 414-4H	1.4 MB	有	有	无	有	64MB	1756-L63
6ES7 417-4HL04-0AB0	CPU 417-4H	20MB	有	有	无	有	64MB	1756-L64
6ES7 416-2FK04-0AB0	CPU-416F-2	2.6MB	有	有	无	无	64MB	1756-L61S

数字输入模块

Siemens 产品目录号	前端连接器	点数	范围	RA 解决方案	备注
6ES7 421-7BH01-0AB0 （中断 / 诊断）	48 针	16	24V DC	1756-IB16D	
6ES7 421-1BL01-0AA0	48 针	32	24V DC	1756-IB32	
6ES7 421-1EL00-0AA0	48 针	32	120V AC/DC	1756-IA32	
6ES7 421-1FH20-0AA0	48 针	16	230V AC/DC	1756-IM161	
6ES7 421-7DH00 0AB0 （中断 / 诊断）	48 针	32	24-60V AC/DC		

数字输出模块

Siemens 产品目录号	前端连接器	点数	范围	电流	RA 解决方案	备注
6ES7 422-1FH00-0AA0	48 针	16	230VAC	2 A	1756-OA16	
6ES7 422-1HH00-0AA0	48 针	16	60V DC230V AC（继电器）	5 A	1756-OW16I	
6ES7 422 1BH11-0AA0	48 针	16	24 VDC	2 A	1756-OB16E	
6ES7 422-1BL00-0AA0	48 针	32	24 VDC	0.5 A	1756-OB32	
6ES7 422-7BL00-0AB0（诊断）	48 针	32	24 VDC	0.5 A	1756-OB16D 1756-OB32	

模拟输入模块

Siemens 产品目录号	前端连接器	通道数	分辨率 （位）	类型	RA 解决方案	备注
6ES7 431-0HH0-0AB0	48 针	16	13	电压 电流	1756-IF16	16 位
6ES7 431-1KF00-0AB0	48 针	8	13	电压 电流 阻抗	1756-IF8	16 位 4 路差分输入
6ES7 431-1KF10-0AB0	48 针	8	14-16	电压 电流 热电偶 热敏电阻 阻抗	1756-IR6I 1756-IT6I	6 RTD 6 热电偶 都为 16 位
6ES7 431-1FK20-0AB0	48 针	8	14	电压 电流 阻抗	1756-IF16	16 位
6ES7 431-7QH00-0AB0（中断）	48 针	16	16	电压 电流 热电偶 热敏电阻 阻抗	1756-IR6I 1756-IT6I	6 RTD 6 热电偶
6ES7 431-7KF00-0AB0	48 针	8	16	电压 电流 热电偶	1756-IT6I	6 通道
6ES7 431-7KF01-0AB0	48 针	8	16	热敏电阻	1756-IR6I	5 通道

模拟输出模块

Siemens 产品目录号	前端连接器	通道数	分辨率 （位）	类型	RA 解决方案	备注
6ES7 432-1HF00-0AB0	48 针	8	13	电压 电流	1756-OF8	15 位

Siemens HMI 对照表

使用本附录可将罗克韦尔自动化面板与特定类型的 Siemens 面板进行比较。

SIMATIC Micro 面板和罗克韦尔自动化同等产品

SIMATIC Micro 面板					罗克韦尔自动化解决方案		
Siemens 产品目录号	简短参考号	说明	内存	通讯选件	罗克韦尔自动化产品目录号	名称	说明
6AV6640-0BA11-0AX0	SIMATIC OP 73MICRO	3 英寸 STN 单色显示屏，160x48 像素，键盘，仅限 24V DC	128 KB	1xRS485，兼容 S7-200，无打印机端口	2711P-K4M5D	PanelView Plus 400 灰度键盘	3.8 英寸 STN 32 级灰度显示屏，320 x 240 像素，RS-232 通讯，键盘，24V DC，64 MB 闪存，USB 打印功能
6AV6545-0AA15-2AX0	SIMATIC TP070 2007 年 4 月已淘汰	5.7 英寸 STN 显示屏，蓝色模式（4 级），320x240 像素，触摸式，仅限 24V DC	128 KB	1xRS485，兼容 S7-200，无打印机端口	2711P-T6M5D	PanelView Plus 600 灰度触摸屏	5.5 英寸 STN 32 级灰度显示屏，320 x 240 像素，RS-232 通讯，触摸屏，24V DC，USB 打印功能
6AV6640-0CA01-0AX0	SIMATIC TP 170MICRO 2007 年 4 月已淘汰	5.7 英寸 STN 显示屏，蓝色模式（4 级），320x240 像素，触摸屏，仅限 24V DC，有限应用功能	256KB	1xRS485，兼容 S7-200，无打印机端口	2711P-T6M5D	PanelView Plus 600 灰度触摸屏	5.5 英寸 STN 32 级灰度显示屏，320x240 像素，RS-232 通讯，触摸屏，24V DC，USB 打印功能

SIMATIC Micro 面板					罗克韦尔自动化解决方案		
Siemens 产品目录号	简短参考号	说明	内存	通讯选件	罗克韦尔自动化产品目录号	名称	说明
6AV6640-0CA11-0AX0	SIMATIC TP 177MICRO	5.7 英寸 STN 显示屏，蓝色模式（4 级），320x240 像素，触摸屏，仅限 24V DC	256KB	1xRS485，兼容 S7-200，无打印机端口	2711P-T6M5D	PanelView Plus 600 灰度触摸屏	5.5 英寸 STN 32 级灰度显示屏，320x240 像素，RS-232 通讯，触摸屏，24V DC，USB 打印功能
6AV6610-0AA01-1CA8	WINCC FLEXIBLE MICRO 软件	只适用于 Simatic Micro 面板的配置和编程软件	无	无	9701-VWSTMENE	RSView Studio Machine Edition 软件	RSView Studio for Machine Edition 配置软件，用于开发和测试机器级 HMI 应用程序

SIMATIC 面板 - 7*x* 系列和罗克韦尔自动化同等产品

SIMATIC 面板 - 7*x* 系列					罗克韦尔自动化解决方案		
Siemens 产品目录号	简短参考号	说明	内存	通讯选件	罗克韦尔自动化产品目录号	名称	说明
6AV6641-0AA11-0AX0	SIMATIC OP73	3 英寸 STN 单色显示屏，160x48 像素，键盘，仅限 24V DC	256KB	1x RS485，S7-200，S7- 300/400 兼容，无打印机端口	2711P-K4M5D	PanelView Plus 400 灰度键盘	3.8 英寸 STN 32 级灰度显示屏，320x240 像素，RS-232 通讯，键盘，24V DC，64 MB 闪存，USB 打印功能
6AV6641-0BA11-0AX0	SIMATIC OP77A	4.5 英寸 STN 单色显示屏，160x64 像素，键盘。仅限 24V DC	256KB	1xRS422，1xRS485，S7-200，S7-300/400，无打印机端口	2711P-K4M5D	PanelView Plus 400 灰度键盘	3.8 英寸 STN 32 级灰度显示屏，320x240 像素，RS-232 通讯，键盘，24V DC，64 MB 闪存，USB 打印功能
6AV6641-0CA01-0AX0	SIMATIC OP77B	4.5 英寸 STN 单色显示屏，160x64 像素，键盘，仅限 24V DC	1 MB	1xRS232，1xRS422，1xRS485，USB，S7-200，S7-300/400，有打印机端口	2711P-K4M5D	PanelView Plus 400 灰度键盘	3.8 英寸 STN 32 级灰度显示屏，320x240 像素，RS-232 通讯，键盘，24V DC，64 MB 闪存，USB 打印功能

SIMATIC 面板 - 7*x* 系列					罗克韦尔自动化解决方案		
Siemens 产品目录号	简短参考号	说明	内存	通讯选件	罗克韦尔自动化产品目录号	名称	说明
6AV6621-0AA01-0AA0	WINCC FLEXIBLE COMPACT 软件	Simatic OP77、OP/TP170 和 Micro 面板的配置和编程软件	无	无	9701-VWSTMENE	RSView Studio for Machine Edition 软件	RSView Studio Machine Edition 配置软件，用于开发和测试机器级 HMI 应用程序

SIMATIC 面板 - 17*x* 系列和罗克韦尔自动化同等产品

SIMATIC 面板 - 17*x* 系列					罗克韦尔自动化解决方案		
Siemens 产品目录号	简短参考号	说明	内存	通讯选件	罗克韦尔自动化产品目录号	名称	说明
6AV6545-0BA15-2AX0	SIMATIC TP170A 蓝色模式 2007 年 4 月已淘汰	5.7 英寸 STN 显示屏，蓝色模式（4 级），320x240 像素，触摸屏，仅限 24V DC	320 KB	1xRS232，1xRS422，1xRS485，S5，S7-200，S7-300/400，第三方控制器，无打印机端口	2711P-T6M20D	PanelView Plus 600 灰度触摸屏	5.5 英寸 STN 32 级灰度显示屏，320x240 像素，EtherNet/IP，RS-232 通讯，触摸屏，24V DC，64 MB 闪存，USB 打印功能
6AV6545-0BB15-2AX0	SIMATIC TP170B 蓝色模式 2007 年 4 月已淘汰	5.7 英寸 STN 显示屏，蓝色模式（4 级），320x240 像素，触摸屏，仅限 24V DC	768 KB	2xRS232，1xRS422，1xRS485，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-T6M20D	PanelView Plus 600 灰度触摸屏	5.5 英寸 STN 32 级灰度显示屏，320x240 像素，EtherNet/IP，RS-232 通讯，触摸屏，24V DC，64 MB 闪存，USB 打印功能
6AV6545-0BC15-2AX0	SIMATIC TP170B 彩色 2007 年 4 月已淘汰	5.7 英寸 STN 显示屏，彩色（256 色），320x240 像素，触摸屏。仅限 24V DC	768 KB	2xRS232，1xRS422，1xRS485，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-T6C20D	PanelView Plus 600 彩色触摸屏	5.5 英寸 TFT 彩色显示屏，320x240 像素，18 位色深，EtherNet/IP，RS-232 通讯，触摸屏，24V DC，64 MB 闪存，USB 打印功能

SIMATIC 面板 - 17*x* 系列					罗克韦尔自动化解决方案		
Siemens 产品目录号	**简短参考号**	**说明**	**内存**	**通讯选件**	**罗克韦尔自动化产品目录号**	**名称**	**说明**
6AV6542-0BB15-2AX0	SIMATIC OP170B 蓝色模式 2007 年 4 月已淘汰	5.7 英寸 STN 显示屏，蓝色模式（4 级），320x240 像素，键盘和触摸屏，仅限 24V DC	768 KB	2xRS232，1xRS422，1xRS485，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-B6M20D	PanelView Plus 600 灰度触摸屏和键盘	5.5 英寸 STN 32 级灰度显示屏，320x240 像素，EtherNet/IP，RS-232 通讯，触摸屏和键盘，24V DC，64 MB 闪存，USB 打印功能
6AV6642-0DC01-1AX0	SIMATIC OP177B 蓝色模式	5.7 英寸 STN 显示屏，蓝色模式（4 级），320x240 像素，键盘和触摸屏，仅限 24V DC	2 MB	1xRS422，1xRS485，USB，Ethernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-B6M20D	PanelView Plus 600 灰度触摸屏和键盘	5.5 英寸 STN 32 级灰度显示屏，320x240 像素，EtherNet/IP，RS-232 通讯，触摸屏和键盘，24V DC，64 MB 闪存，USB 打印功能
6AV6642-0AA11-0AX0	SIMATIC TP177A 蓝色模式	5.7 英寸 STN 显示屏，蓝色模式（4 级），320x240 像素，触摸屏，仅限 24V DC	512 KB	1xRS422，1xRS485，S7-200，S7-300/400 兼容，无打印机端口	2711P-T6M20D	PanelView Plus 600 灰度触摸屏	5.5 英寸 STN 32 级灰度显示屏，显示 320 x 240 像素，EtherNet/IP，RS-232 通讯，触摸屏，24VDC，64 MB 闪存，USB 打印功能
6AV6642-0BA01-1AX0	SIMATIC TP177B 彩色	5.7 英寸 STN 显示屏，彩色（256 色），320x240 像素，触摸屏。仅限 24V DC	2 MB	1xRS422，1xRS485，USB，Ethernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-T6C20D	PanelView Plus 600 彩色触摸屏	5.5 英寸 TFT 彩色显示屏，320 x 240 像素，18 位色深，EtherNet/IP，RS-232 通讯，触摸屏，24VDC，64 MB 闪存，USB 打印功能
6AV6642-0BC01-1AX0	SIMATIC TP177B 蓝色模式	5.7 英寸 STN 显示屏，蓝色模式（4 级），320x240 像素，触摸屏，仅限 24V DC	2 MB	1xRS422，1xRS485，USB，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-T6M20D	PanelView Plus 600 灰度触摸屏	5.5 英寸 STN 32 级灰度显示屏，显示 320 x 240 像素，EtherNet/IP，RS-232 通讯，触摸屏，24VDC，64 MB 闪存，USB 打印功能

SIMATIC 面板 - 17*x* 系列					罗克韦尔自动化解决方案		
Siemens 产品目录号	**简短参考号**	**说明**	**内存**	**通讯选件**	**罗克韦尔自动化产品目录号**	**名称**	**说明**
6AV6642-8BA10-0AA0	SIMATIC TP177B 彩色不锈钢	5.7 英寸 STN 显示屏，彩色（256 色），320x240 像素，触摸屏，仅限 24V DC，不锈钢面板边框	2 MB	1xRS422，1xRS485，USB，Ethernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-T6C20D	PanelView Plus 600 彩色触摸屏	5.5 英寸 TFT 彩色显示屏，320x240 像素，18 位色深，EtherNet/IP，RS-232 通讯，触摸屏，24V DC，64 MB 闪存，USB 打印功能
6AV6642-0DA01-1AX0	SIMATIC OP177B 彩色	5.7 英寸 STN 显示屏，彩色（256 色），320x240 像素，键盘和触摸屏，仅限 24V DC	2 MB	1xRS422，1xRS485，USB，Ethernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-B6C20D	PanelView Plus 600 彩色触摸屏和键盘	5.5 英寸 TFT 彩色显示屏，320x240 像素，18 位色深，EtherNet/IP，RS-232 通讯，触摸屏和键盘，24V DC，64 MB 闪存，USB 打印功能
6AV6621-0AA01-0AA0	WINCC FLEXIBLE COMPACT 软件	适用于 Simatic OP77、OP/TP170 和 Micro 面板的配置和编程软件	无	无	9701-VWSTMENE	RSView Studio Machine Edition 软件	RSView Studio Machine Edition 配置软件，用于开发和测试机器级 HMI 应用程序

SIMATIC 面板 - 27x 系列和罗克韦尔自动化同等产品

SIMATIC 面板 - 27*x* 系列					罗克韦尔自动化解决方案		
Siemens 产品目录号	简短参考号	说明	内存	通讯选件	罗克韦尔自动化产品目录号	名称	说明
6AV6545-0CA10-0AX0	SIMATIC TP270 6 英寸彩色 2006 年 10 月已淘汰	5.7 英寸 STN 显示屏，彩色（256 色），320x240 像素，触摸屏，仅限 24V DC	2 MB	2xRS232，1xRS422，1xRS485，USB，S5，S7-200，S7-300/400，第三方控制器，有打印机端口。	2711P-T6C20D	PanelView Plus 600 彩色触摸屏	5.5 英寸 TFT 彩色显示屏，320x240 像素，18 位色深，EtherNet/IP，RS-232 通讯，触摸屏，24V DC，64 MB 闪存，USB 打印功能
6AV6545-0CC10-0AX0	SIMATIC TP270 10 英寸彩色 2006 年 10 月已淘汰	10.4 英寸 STN 显示屏，彩色（256 色），640x480 像素，触摸屏，仅限 24V DC	2 MB	2xRS232，1xRS422，1xRS485，USB，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-T10C4D1	PanelView Plus 1000 彩色触摸屏	10.4 英寸 TFT 显示屏，640x480 像素，18 位彩色，EtherNet/IP 和 RS-232，触摸屏，24V DC，64 MB 闪存，USB 打印功能
6AV6542-0CA10-0AX0	SIMATIC OP270 6 英寸彩色 2006 年 10 月已淘汰	5.7 英寸 STN 显示屏，彩色（256 色），320x240 像素，键盘，仅限 24V DC	2 MB	2xRS232，1xRS422，1xRS485，USB，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-K6C20D	PanelView Plus 600 彩色	5.5 英寸 TFT 彩色显示屏，320x240 像素，18 位色深，EtherNet/IP，RS-232 通讯，键盘，24V DC，64 MB 闪存，USB 打印功能
6AV6542-0CC10-0AX0	SIMATIC OP270 10 英寸彩色 2006 年 10 月已淘汰	10.4 英寸 STN 显示屏，彩色（256 色），640x480 像素，键盘，仅限 24V DC	2 MB	2xRS232，1xRS422，1xRS485，USB，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-K10C4D1	PanelView Plus 1000 彩色键盘	10.4 英寸 TFT 显示屏，640x480 像素，18 位彩色，EtherNet/IP 和 RS-232，键盘，24V DC，64 MB 闪存，USB 打印功能

SIMATIC 面板 – 27*x* 系列					**罗克韦尔自动化解决方案**		
Siemens 产品目录号	**简短参考号**	**说明**	**内存**	**通讯选件**	**罗克韦尔自动化产品目录号**	**名称**	**说明**
6AV6643-0AA01-1AX0	SIMATIC TP 277 6 英寸彩色	5.7 英寸 STN 显示屏，彩色（256 色），320x240 像素，触摸屏，仅限 24V DC	4 MB	1xRS422，1xRS485，USB，Ethernet：S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-T6C20D	PanelView Plus 600 彩色触摸屏	5.5 英寸 TFT 彩色显示屏，320x240 像素，18 位色深，EtherNet/IP，RS-232 通讯，触摸屏，24V DC，64 MB 闪存，USB 打印功能
6AV6643-0BA01-1AX0	SIMATIC OP 277 6 英寸彩色	5.7 英寸 STN 显示屏，彩色（256 色），320x240 像素，键盘，仅限 24V DC	4 MB	1xRS422，1xRS485，USB，Ethernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口。	2711P-K6C20D	PanelView Plus 600 彩色	5.5 英寸 TFT 彩色显示屏，320x240 像素，18 位色深，EtherNet/IP，RS-232 通讯，键盘，24V DC，64 MB 闪存，USB 打印功能
6AV6622-0BA01-0AA0	WINCC FLEXIBLE STANDARD 软件	适用于 Simatic OP/TP/MP270、MP370、OP77、OP/TP170 和 micro 面板的配置和编程软件	无	无	9701-VWSTMENE	RSView Studio Machine Edition 软件	RSView Studio Machine Edition 配置软件，用于开发和测试机器级 HMI 应用程序

SIMATIC 多面板 - 27*x* 系列和罗克韦尔自动化同等产品

SIMATIC 多面板 - 27*x* 系列					罗克韦尔自动化解决方案		
Siemens 产品目录号	简短参考号	说明	内存	通讯选件	罗克韦尔自动化产品目录号	名称	说明
6AV6542-0AG10-0AX0	SIMATIC MP270B 键盘 10 英寸 2006 年 10 月已淘汰	10.4 英寸 TFT 显示屏，彩色 （64 k 色），640x480 像素，键盘，仅限 24V DC	5 MB	2xRS422，1xRS485，USB，Ethernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-K10C4D1	PanelView Plus 1000 彩色键盘	10.4 英寸 TFT 显示屏，640x480 像素，18 位彩色，EtherNet/IP 和 RS-232，键盘，24V DC，64 MB 闪存，USB 打印功能
6AV6545-0AG10-0AX0	SIMATIC MP270B 触摸屏，10 英寸 2006 年 10 月已淘汰	10.4 英寸 TFT 显示屏，彩色 （64 k 色），640x480 像素，触摸屏，仅限 24V DC	5 MB	2xRS422，1xRS485，USB，Ethernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-T10C4D1	PanelView Plus 1000 彩色触摸屏	10.4 英寸 TFT 显示屏，640x480 像素，18 位彩色，EtherNet/IP 和 RS-232，触摸屏，24V DC，64 MB 闪存，USB 打印功能
6AV6545-0AH10-0AX0	SIMATIC MP270B 触摸屏，6 英寸 2006 年 10 月已淘汰	5.7 英寸TFT 显示屏，彩色（64 k 色），320x240 像素，触摸屏，仅限 24V DC	5 MB	2xRS422，1xRS485，USB，Ethernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-K6C20D	PanelView Plus 600 彩色	5.5 英寸 TFT 彩色显示屏，320x240 像素，18 位色深，EtherNet/IP，RS-232 通讯，键盘，24V DC，64 MB 闪存，USB 打印功能
6AV6643-0CB01-1AX0	SIMATIC MP 277 触摸屏，8 英寸	7.5 英寸 TFT 显示屏，彩色 （64 k 色），640x480 像素，触摸屏，仅限 24V DC	6 MB	1xRS422，1xRS485，2xUSB，Ethernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-T7C4D1	PanelView Plus 700 彩色触摸屏	6.5 英寸 TFT 显示屏，640x480 像素，18 位彩色，EtherNet/IP 和 RS-232，触摸屏，24V DC，64 MB 闪存，USB 打印功能
6AV6643-0CD01-1AX0	SIMATIC MP 277 触摸屏，10 英寸	10.4 英寸 TFT 显示屏，彩色 （64 k 色），640x480 像素，触摸屏，仅限 24V DC	6 MB	1xRS422，1xRS485，2xUSB，Ethernet：S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-T10C4D1	PanelView Plus 1000 彩色触摸屏	10.4 英寸 TFT 显示屏，640x480 像素，18 位彩色，EtherNet/IP 和 RS-232，触摸屏，24V DC，64 MB 闪存，USB 打印功能
–	SIMATIC MP 277 触摸屏，10 英寸，不锈钢	10.4 英寸 TFT 显示屏，彩色 （64 k 色），640x480 像素，触摸屏，仅限 24V DC，不锈钢面板边框，IP66	6 MB	1xRS422，1xRS485，2xUSB，Ethernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-T10C4D1	PanelView Plus 1000 彩色触摸屏	10.4 英寸 TFT 显示屏，640x480 像素，18 位彩色，EtherNet/IP 和 RS-232，触摸屏，24V DC，64 MB 闪存，USB 打印功能

SIMATIC 多面板 - 27*x* 系列					罗克韦尔自动化解决方案		
Siemens 产品目录号	**简短参考号**	**说明**	**内存**	**通讯选件**	**罗克韦尔自动化产品目录号**	**名称**	**说明**
6AV6643-0DB01-1AX0	SIMATIC MP 277 键盘，8 英寸	7.5 英寸 TFT 显示屏，彩色 （64 k 色），640x480 像素，键盘，仅限 24V DC	6 MB	1xRS422，1xRS485，2xUSB，Ethernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-K7C4D1	PanelView Plus 700 彩色键盘	6.5 英寸 TFT 显示屏，640x480 像素，18 位彩色，EtherNet/IP 和 RS-232，键盘，24V DC，64 MB 闪存，USB 打印功能
6AV6643-0DD01-1AX0	SIMATIC MP 277 键盘，10 英寸	10.5 英寸 TFT 显示屏，彩色 （64 k 色），640x480 像素，键盘，仅限 24V DC	6 MB	1xRS422，1xRS485，2xUSB，Ethernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口	2711P-K10C4D1	PanelView Plus 1000 彩色键盘	10.4 英寸 TFT 显示屏，640x480 像素，18 位彩色，EtherNet/IP 和 RS-232，键盘，24V DC，64 MB 闪存，USB 打印功能
6AV6622-0BA01-0AA0	WINCC FLEXIBLE STANDARD 软件	适用于 Simatic OP/TP/MP270、MP370、OP77、OP/TP170 和 micro 面板的配置和编程软件	无	无	9701-VWSTMENE	RSView Studio Machine Edition 软件	RSView Studio Machine Edition 配置软件，用于开发和测试机器级 HMI 应用程序

SIMATIC 多面板 - 37*x* 系列和罗克韦尔自动化同等产品

SIMATIC 多面板 - 37*x* 系列					罗克韦尔自动化解决方案		
Siemens 产品目录号	简短参考号	说明	内存	通讯选件	罗克韦尔自动化产品目录号	名称	说明
6AV6542-0DA10-0AX0	SIMATIC MP370 键盘，12 英寸	12.1 英寸 TFT 显示屏，彩色（256 色），800x600 像素，键盘，仅限 24V DC	12.5 MB	1xTTY，2xRS232，1xRS422，1xRS485，1xUSB，Ethernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口。	2711P-K12C4D1	PanelView Plus 1250 彩色键盘	12.1 英寸 TFT 显示屏，800x600 像素，18 位彩色，EtherNet/IP 和 RS-232，键盘，24V DC，64 MB 闪存，USB 打印功能
6AV6545-0DA10-0AX0	SIMATIC MP370 触摸屏，12 英寸	12.1 英寸 TFT 显示屏，彩色（256 色），800x600 像素，触摸屏，仅限 24V DC	12.5 MB	1xTTY，2xRS232，1xRS422，1xRS485，1xUSB，Ethernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口。	2711P-T12C4D1	PanelView Plus 1250 彩色触摸屏	12.1 英寸 TFT 显示屏，800x600 像素，18 位彩色，EtherNet/IP 和 RS-232，触摸屏，24V DC，64 MB 闪存，USB 打印功能
6AV6545-0DB10-0AX0	SIMATIC MP370 触摸屏，15 英寸	15.1 英寸 TFT 显示屏，彩色（256 色），1024x768 像素，触摸屏，仅限 24V DC	12.5 MB	1xTTY，2xRS232，1xRS422，1xRS485，1xUSB，Ethernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口。	2711P-T15C4D1	PanelView Plus 1500 彩色触摸屏	15 英寸 TFT 显示屏，1024x768 像素，18 位彩色，EtherNet/IP 和 RS-232，触摸屏，24V DC，64 MB 闪存，USB 打印功能
6AV6545-8DB10-0AA0	SIMATIC MP370 触摸屏，15 英寸，不锈钢	15.1 英寸 TFT 显示屏，彩色（256 色），1024x768 像素，触摸屏，仅限 24V DC，不锈钢面板边框，IP66	12.5 MB	1xTTY，2xRS232，1xRS422，1xRS485，1xUSB，Ethernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口。	2711P-T15C4D1	PanelView Plus 1500 彩色触摸屏	15 英寸 TFT 显示屏，1024x768 像素，18 位彩色，EtherNet/IP 和 RS-232，触摸屏，24V DC，64 MB 闪存，USB 打印功能
6AV6 644-0AA01-2AX0	SIMATIC MP377 触摸屏 12.1 英寸	12.1 英寸 TFT 显示屏，65,536 色，800x600 像素，触摸屏，仅限 24V DC	12.5 MB	1xTTY，2xRS232，1xRS422，1xRS485，2xUSB，2xEthernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口。	2711P-T12C4D1	PanelView Plus 1250 彩色触摸屏	12.1 英寸 TFT 显示屏，800x600 像素，18 位彩色，EtherNet/IP 和 RS-232，触摸屏，24V DC，64 MB 闪存，USB 打印功能

SIMATIC 多面板 – 37*x* 系列					**罗克韦尔自动化解决方案**		
Siemens 产品目录号	**简短参考号**	**说明**	**内存**	**通讯选件**	**罗克韦尔自动化产品目录号**	**名称**	**说明**
6AV6 644-0BA01-2AX0	SIMATIC MP377 键盘，12.1 英寸	12.1 英寸 TFT 显示屏，65,536 色，800x600 像素，键盘，仅限 24V DC	12.5 MB	1xTTY，2xRS232，1xRS422，1xRS485，2xUSB，2xEthernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口。	2711P-K12C4D1	PanelView Plus 1250 彩色键盘	12.1 英寸 TFT 显示屏，800x600 像素，18 位彩色，EtherNet/IP 和 RS-232，键盘，24V DC，64 MB 闪存，USB 打印功能
6AV6 644-0AB01-2AX0	SIMATIC MP377 触摸屏，15 英寸	15 英寸 TFT 显示屏，65,536 色，1024x768 像素，触摸屏，仅限 24V DC	12.5 MB	1xTTY，2xRS232，1xRS422，1xRS485，2xUSB，2xEthernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口。	2711P-T15C4D1	PanelView Plus 1500 彩色触摸屏	15 英寸 TFT 显示屏，1024x768 像素，18 位彩色，EtherNet/IP 和 RS-232，触摸屏，24V DC，64 MB 闪存，USB 打印功能
6AV6 644-0BA01-2AX0	SIMATIC MP377 触摸屏，19 英寸	19 英寸 TFT 显示屏，65,536 色，1280x1024 像素，触摸屏，仅限 24V DC	12.5 MB	1xTTY，2xRS232，1xRS422，1xRS485，2xUSB，2xEthernet，S5，S7-200，S7-300/400，第三方控制器，有打印机端口。	2711P-T15C4D1	PanelView Plus 1500 彩色触摸屏	15 英寸 TFT 显示屏，1024x768 像素，18 位彩色，EtherNet/IP 和 RS-232，触摸屏，24V DC，64 MB 闪存，USB 打印功能
6AV6622-0BA01-0AA0	WINCC FLEXIBLE STANDARD 软件	Simatic OP/TP/MP270、MP370、OP77、OP/TP170 和 micro 面板的配置和编程软件	无	无	9701-VWSTMENE	RSView Studio Machine Edition 软件	RSView Studio Machine Edition 配置软件，用于开发和测试机器级 HMI 应用程序

注:

罗克韦尔自动化支持

罗克韦尔自动化在 Web 上提供技术信息，以帮助您使用产品。在 http://support.rockwellautomation.com 上，您可以找到技术手册、常见问题知识库、技术和应用说明、示例代码以及指向软件服务包的链接，而且您还可以通过自定义 MySupport 功能，充分发挥这些工具的作用。

为针对安装、配置和故障排除提供更高级别的电话技术支持，我们推出了 TechConnect 支持计划。有关更多信息，请联系当地经销商或罗克韦尔自动化代表，或访问 http://support.rockwellautomation.com。

安装协助

如果在安装后 24 小时内遇到问题，请查阅本手册中的信息。您也可以拨打专门的客户支持号码，以获启动和运行产品的初始安装帮助。

美国	1.440.646.3434 周一至周五，东部时间早上 8 点至下午 5 点
美国之外	如有任何技术支持问题，请联系当地的罗克韦尔自动化代表。

新产品不满意退货

罗克韦尔自动化的所有产品均经过检测，以确保在出厂时能够正常工作。但是，如果您购买的产品不能正常工作并且需要退货，请按照以下程序进行退货。

美国	联系您的经销商。要完成退货过程，您必须向经销商提供客户支持案例号 （请拨打上面的电话号码以获得案例号）。
美国之外	请联系当地的罗克韦尔自动化代表，了解退货程序。

www.rockwellautomation.com

动力，控制与信息解决方案

美国: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, 电话: (1) 414.382.2000, 传真: (1) 414.382.4444
欧洲/中东/非洲地区: Rockwell Automation, Vorstlaan/Boulevard du Souverain 36, 1170 Brussels, Belgium, 电话: (32) 2 663 0600, 传真: (32) 2 663 0640
亚太地区: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, 电话: (852) 2887 4788, 传真: (852) 2508 1846

北京 - 北京市建国门内大街18号恒基中心办公楼1座4层 邮编：100005 电话：(8610)65182535 传真：(8610)65182536 www.rockwellautomation.com.cn
青岛 - 青岛市香港中路40号数码港旗舰大厦2206室 邮编：266071 电话：(86532)86678338 传真：(86532)86678339
西安 - 西安市高新区科技路33号高新国际商务中心数码大厦1201,1202,1208室 邮编：710075 电话：(8629)88152488 传真：(8629)88152466
郑州 - 郑州市中原中路220号裕达国际贸易中心A座1216-1218室 邮编：450007 电话：(86371)67803366 传真：(86371)67803388
上海 - 上海市仙霞路319号远东国际广场A幢7楼 邮编：200051 电话：(8621)61206007 传真：(8621)62351099
南京 - 南京市中山南路49号商茂世纪广场44楼A3-A4座 邮编：210005 电话：(8625)86890445 传真：(8625)86890142
武汉 - 武汉市建设大道568号新世界国贸大厦I座2202室 邮编：430022 电话：(8627)68850233 传真：(8627)68850232
广州 - 广州市环市东路362号好世界广场2703-04室 邮编：510060 电话：(8620)83849977 传真：(8620)83849989
深圳 - 深圳市深南东路5047号深圳发展银行大厦15L 邮编：518001 电话：(86755)25847099 传真：(86755)25870900
厦门 - 厦门市湖里区湖里大道41号联泰大厦4A单元西侧 邮编：361006 电话：(86592)2655888 传真：(86592)2655999
成都 - 成都市总府路2号时代广场A座906室 邮编：610016 电话：(8628)86726886 传真：(8628)68726887
重庆 - 重庆市渝中区邹容路68号大都会商厦3112-13室 邮编：400010 电话：(8623)63702668 传真：(8623)63702558
沈阳 - 沈阳市沈河区青年大街219号华新国际大厦15-F单元 邮编：110015 电话：(8624)23961518 传真：(8624)23963539
大连 - 大连市西岗区中山路147号森茂大厦2305层 邮编：116011 电话：(86411)83687799 传真：(86411)83679970
哈尔滨 - 哈尔滨市南岗区红军街15号奥威斯发展大厦七层E座 邮编：150001 电话：(86451)84879066 传真：(86451)84879088

出版号 LOGIX-AP008B-ZH-P - 6 月 2008
替代出版号 LOGIX-AP008B-ZH-P