SIMATIC S7-1200/1500
编程指南

TIA Portal

https://support.industry.siemens.com/cs/cn/zh/view/90885040

西门子工业在线支持

法律信息

应用实例的使用

应用实例说明了通过文本、图形和/或软件模块形式的几个组件的交互来实现自动化任务的解决方案。应用实例是由西门子公司和/或西门子公司的子公司（“西门子”）提供的免费服务。它们是非约束性的，并且不对配置和设备的完整性或功能性做出任何声明。应用程序示例仅对典型任务提供帮助;它们不构成针对客户的解决方案。您自己有责任按照适用的规定正确和安全地操作产品，并必须检查相应应用示例的功能，并为您的系统定制应用示例。
西门子授予您非排他性、不可再授权和不可转让的权利，让经过技术培训的人员使用应用示例。对应用程序示例的任何更改均由您负责。与第三方共享应用程序示例或复制应用程序示例或摘录，只有在与您自己的产品组合时才允许。应用实例不要求经过收费产品的惯常测试和质量检验，它们可能有功能和性能缺陷以及错误。您有责任以可能发生的任何故障不会导致财产损失或人员伤害的方式使用它们。

免责声明

无论何种法律原因，西门子均不承担任何责任，包括但不限于对应用示例的可用性、有效性、完整性和无缺陷以及相关信息、配置和性能数据以及由此造成的任何损害承担任何责任。这个不适用强制责任的情况下，例如，根据《德国产品责任法》，或在存在故意、重大过失、应负责任的生命损失、身体伤害或健康损害、不遵守保证、欺诈性不披露缺陷或应负责任的重大合同义务的情况下。但是，因违反重大合同义务而引起的损害索赔应限于典型协议类型的可预见损害，除非责任是由于故意或重大过失或基于生命损失、身体伤害或健康损害而引起的。上述条款并不意味着对损害贵方利益的举证责任有任何改变。除非西门子被强制承担责任，否则贵方应向西门子赔偿目前或未来第三方就此提出的索赔。

通过使用应用实例，您承认西门子不承担超出上述责任条款的任何损害。

其他信息

西门子保留在不另行通知的情况下随时对应用实例进行更改的权利。如果应用示例中的建议与其他西门子出版物(如目录)之间存在差异，应以其他文件的内容为准。
西门子使用条款(https://support.industry.siemens.com)也应适用。

安全信息

西门子提供工业安全功能的产品和解决方案，支持工厂、系统、机器和网络的安全运行。

为了保护工厂、系统、机器和网络免受网络威胁，有必要实施并持续维护一个整体的、最先进的工业安全概念。西门子的产品和解决方案构成了这一概念的一个要素。

客户有责任防止未经授权访问其工厂、系统、机器和网络。该等系统、机器及组件只应在有必要及有适当的保安措施(例如防火墙及/或网络分段)的情况下，才可连接到企业网络或互联网。

有关可能实施的工业安全措施的更多信息，请访问
https://www.siemens.com/industrialsecurity

西门子的产品和解决方案不断发展，使其更加安全。西门子强烈建议，一旦产品更新可用，就应用最新的产品版本。使用不再受支持的产品版本，以及未能应用最新更新，可能增加客户遭受网络威胁的风险。

要了解产品更新，请订阅西门子工业安全 RSS: https://www.siemens.com/industrialsecurity

目录

1 前言

开发新一代 SIMATIC 控制器的目标

- 适用于所有自动化组件（控制器、HMI、驱动器等）的工程框架
- 统一编程
- 提高性能
- 每种语言的完整命令集
- 完全符号程序生成
- 不需要指针的数据处理
- 创建块的可重用性

指南的目标

新一代控制器 SIMATIC S7-1200 和 S7-1500 具有最新的系统架构，并且与 TIA 博途一起提供了新的高效编程和配置选项。最重要的不再是控制器的资源（例如内存中的数据存储），而是实际的自动化解决方案本身。

本文档为您提供了许多有关 S7-1200/1500 控制器优化编程的建议和注意事项。S7-300/400 系统架构的一些差异以及与此关联的新编程选项以易于理解的方式进行了解释。这有助于您为自动化解决方案创建标准化和优化的编程。

所描述的示例可普遍用于控制器 S7-1200 和 S7-1500。

本编程指南的核心内容

本文档涉及 TIA 博途的以下关键问题：

- S7-1200/1500 创新
  - 编程语言
  - 优化的块
  - 数据类型和指令
- 关于一般编程的建议
  - 操作系统和用户程序
  - 内存概念
  - 符号寻址
  - 库
- 与硬件无关的编程建议
- 关于 TIA 博途中 STEP 7 Safety 的建议
- 最重要的建议概述

优势和好处

应用这些建议和技巧可带来许多优势：

- 强大的用户程序
- 清晰的程序结构
- 直观有效的编程解决方案

更多信息

在对 SIMATIC 控制器进行编程时，程序员的任务是创建尽可能清晰易读的用户程序。每个用户都使用自己的策略，例如，如何命名变量或块或注释方式。程序员的不同理念创建了非常不同的用户程序，只能由各自的程序员解释。

编程风格指南为您提供了一组协调一致的编程规则。例如，为了在 SCL 进行清晰的编程，这些规范描述了统一的变量和块的命名规则。

您可以自由地使用这些规则和建议；它们仅作为标准化编程的建议（不是编程标准）。

注意 S7-1200 和 S7-1500 的编程风格指南可在以下链接中找到：

https://support.industry.siemens.com/cs/ww/en/view/81318674

2 S7-1200/S7-1500 创新

2.1 介绍

一般来说，从 S7-300/400 到 S7-1500 的 SIMATIC 控制器的编程保持不变。有熟知的编程语言，例如 LAD、FBD、STL、SCL 或 GRAPH，以及熟知的块，例如组织块(OB)、函数块(FB)、函数(FC)或数据块(DB)。在 S7-300/400 中创建的程序可以轻松的在 S7-1500 上执行，且现有的 LAD、FBD 和 SCL 程序可以轻松的在 S7-1200 控制器上执行。

此外，还有许多创新可以方便您进行编程并实现功能强大且节省内存的代码。

对于为 S7-1200/1500 控制器执行的程序，我们建议不仅要一个一个地执行它们，也要检查新选项，并在可能的情况下使用它们。通常很少的额外的工作，但将获得一个程序代码包括，

- 最适合新 CPU 的内存和运行时间，
- 更容易理解，
- 并且更容易维护。

注意 有关将 S7-300/S7-400 移植到 S7-1500 的信息，请参见以下条目：

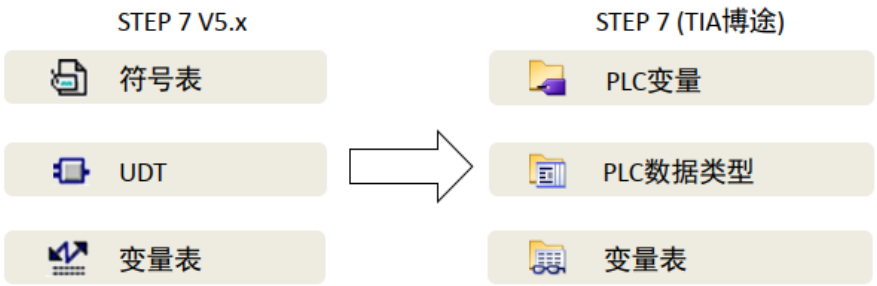
https://support.industry.siemens.com/cs/ww/en/view/109478811

2.2 术语

TIA 博途中的通用术语

某些术语已更改，以便更轻松地使用 TIA 博途进行处理。

图 2 -1：TIA 博途中的新术语



变量和参数的术语

在处理变量、函数和函数块时，许多术语重复使用不同甚至不正确。下图阐明了这些术语。

图 2 -2：变量和参数的术语

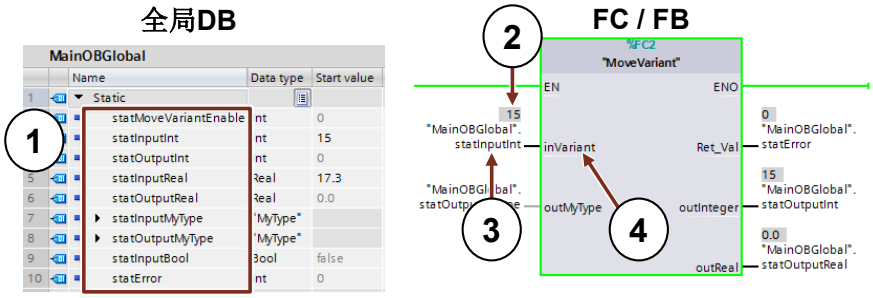


表 2 -3：1

	术语	描述
1.	变量	变量由名称/标识符标记，并使用控制器内存中的地址。变量总是用某种数据类型（布尔、整数等）定义的： • PLC 变量 • 数据块中的单个变量 • 完整的数据块
2.	变量值	变量值是存储在变量中的值（例如，15 是一个整型变量的值）。
3.	实际参数	实参是指令、函数、函数块等接口互联的变量。
4.	形式参数（传递参数、块参数）	形式参数是指令、函数和函数块（Input、Output、InOut 和 Ret_Val）的接口参数。

注意

更多信息可以在以下条目中找到：

互联网上有哪些条目可用于移植到 STEP 7（TIA 博途）和 WinCC（TIA 博途）？
https://support.industry.siemens.com/cs/ww/en/view/56314851

在 STEP 7 Professional（TIA 博途）中移植 STEP 7 V5.x 项目必须满足哪些系统要求？

https://support.industry.siemens.com/cs/ww/en/view/62100731

使用 STEP 7（TIA 博途）将 PLC 移植到 S7-1500
https://support.industry.siemens.com/cs/ww/en/view/67858106

如何在 STEP 7（TIA 博途）中为 S7-1200/S7-1500 高效地编程？
https://support.industry.siemens.com/cs/ww/en/view/67582299

为什么对于 S7-1500 在 STEP 7 (TIA 博途) 中寄存器传递和实参传输不能混用？本条目还介绍了将 STL 程序移植到 S7-1500。
https://support.industry.siemens.com/cs/ww/en/view/67655405

2.3 编程语言

不同的编程语言可用于用户程序的编程。每种语言都有自己的优势，可以根据应用灵活使用。因此，用户程序中的每个块都可以用任何编程语言创建。

表 2 -4：编程语言

编程语言	S7-1200	S7-1500
梯形图 (LAD)	支持	支持
功能块图(FBD)	支持	支持
结构化控制语言(SCL)	支持	支持
Graph	不支持	支持
语句表(STL)	不支持	支持

注意

更多信息可以在以下条目中找到：

SIMATIC S7-1200/S7-1500 基于国际助记符的编程语言比较列表
https://support.industry.siemens.com/cs/ww/en/view/86630375

在 STEP 7（TIA 博途）中移植 S7-SCL 程序时应注意什么？
https://support.industry.siemens.com/cs/ww/en/view/59784005

在 STEP 7（TIA 博途）的 SCL 程序中不能使用哪些指令？
https://support.industry.siemens.com/cs/ww/en/view/58002709

如何在 STEP 7（TIA 博途）中定义 S7-SCL 程序中的常量？
https://support.industry.siemens.com/cs/ww/en/view/52258437

2.4 优化的机器代码

TIA 博途和 S7 -1200/1500 可在每种编程语言中实现优化运行时的性能。所有语言都以相同的方式直接编译成机器代码。

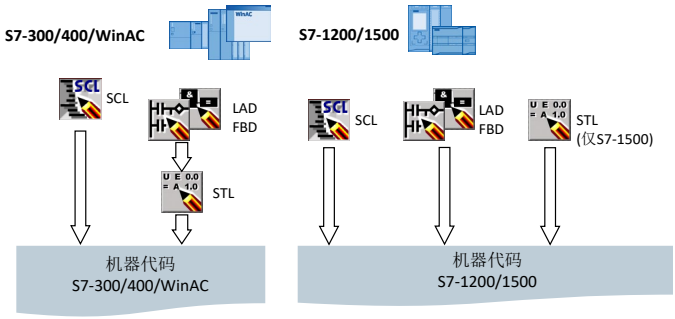
优点

- 所有编程语言都具有相同的性能水平（对于相同的访问类型）
- 对于通过 STL 中间步骤的额外编译不会降低性能

特性

下图展示了 S7-程序在机器代码中编译的不同之处。

图 2 -5：使用 S7-300/400/ WinAC 和 S7-1200/1500 创建机器代码

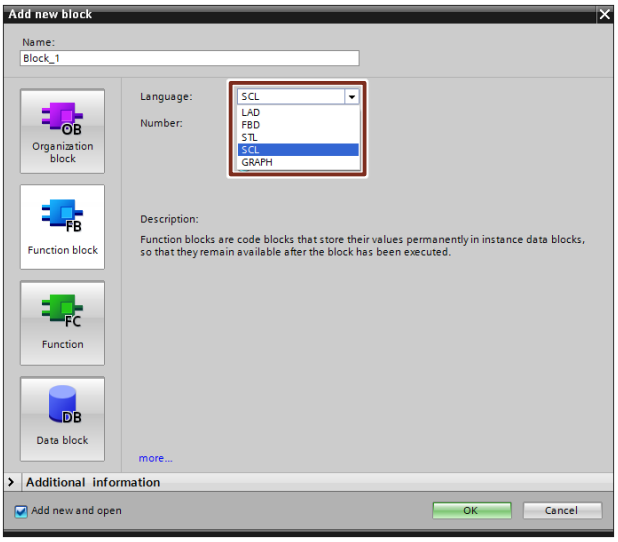


- 对于 S7-300/400/WinAC 控制器，LAD 和 FBD 程序首先在 STL 中编译，然后再创建机器代码。
- 对于 S7-1200/1500 控制器，所有编程语言都直接编译成机器代码。

2.5 创建块

所有块，例如 OB、FB 和 FC，都可以直接用所需的编程语言进行编程。因此，不必为 SCL 编程创建源文件。只需要选择块并将 SCL 作为编程语言，然后可以直接对块进行编程。

图 2 -4：对话框“添加新块”



2.6 优化块

S7-1200/1500 控制器具有优化的数据存储。在优化块中，所有变量都会根据其数据类型自动排序。排序确保变量之间的数据间隙减少到最小，并且变量为处理器的访问优化存储。

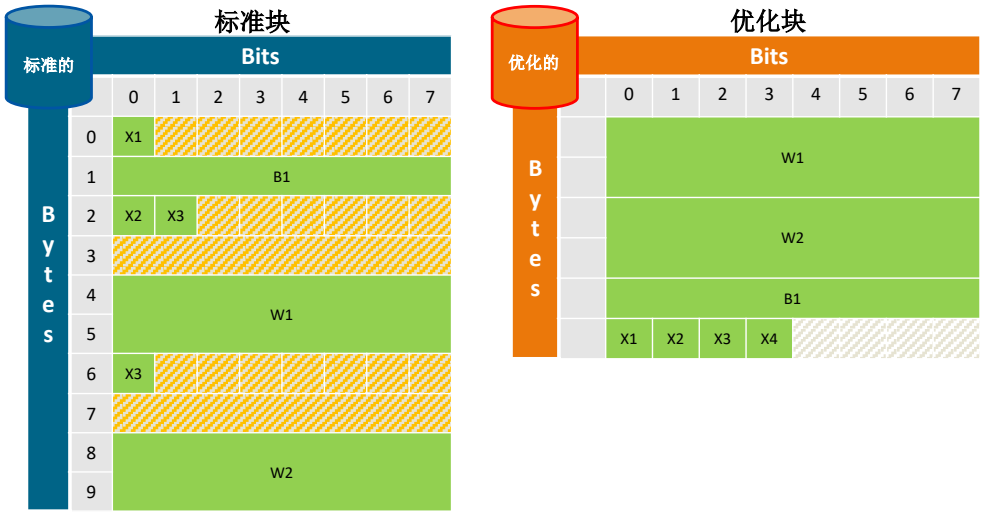
非优化块仅出于兼容性原因用于 S7-1200/1500 控制器中。

优点

- 访问总是尽可能快地进行，因为数据存储由系统优化并且不依赖于声明。
- 没有由于错误的绝对访问而导致不一致的危险，因为访问通常是符号化的
- 声明更改不会导致访问错误，因为例如 HMI 访问是符号化的。
- 单个变量可以单独定义为保持变量。
- 背景数据块中不需要设置。一切都在分配的 FB 中设置（例如，保持性）。
- 数据块中的存储保护区可以在不丢失当前值的情况下进行更改（参见第 3.2.8 章 下载无需重新初始化）。

2.6.1 S7-1200：优化块的结构

图 2 -6：S7-1200 的优化块



特性

- 由于较大的变量位于块的开头，较小的变量位于块的末尾，因此不会形成数据间隙。
- 优化块只有符号访问。

2.6.2 S7-1500：优化块的结构

图 2 -7：S7-1500 的优化块

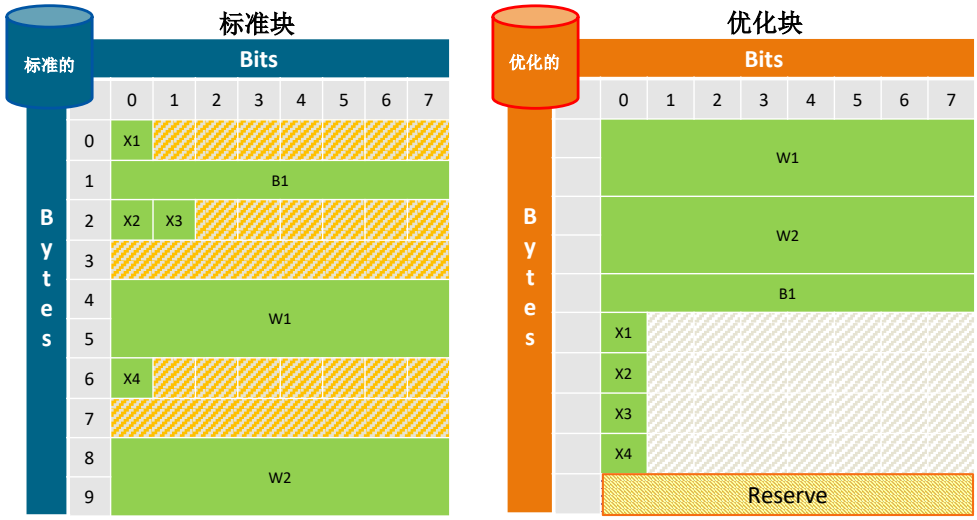
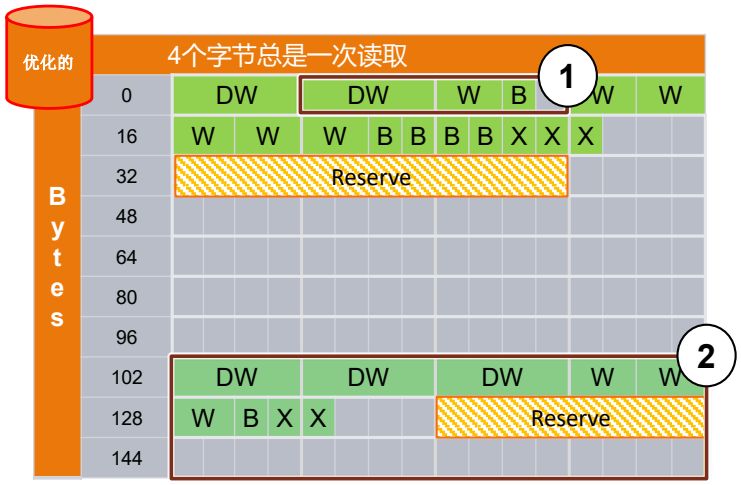


图 2 -8：优化块的内存映射



1. 结构是分开放置的，因此可以作为块复制。
2. 保持性数据位于单独的区域中，可以作为块复制。
   在电压丢失的情况下，该数据会在 CPU 内保存。“MRES”将此数据重置为装载存储器中的起始值。

特性

- 由于较大的变量位于块的开头，较小的变量位于块的末尾，因此不会形成数据间隙。
- 由于处理器优化存储，访问速度更快（所有变量都以某种方式存储，以便 S7-1500 的处理器只需一个机器命令即可直接读取或写入它们）。
- 布尔变量以字节形式存储以便更快地访问。因此，控制器不必以掩码方式访问。
- 优化块有一个存储预留区用于在运行操作中加载（参见第 3.2.8 章 下载无需重新初始化）。
- 优化块只有符号访问。

2.6.3 适用于 S7-1500 的处理器优化数据存储

出于与第一代 SIMATIC 控制器兼容的原因，S7-300/400 控制器接受了“大端”数据存储原则。

基于改变的处理器架构，新一代 S7-1500 控制器始终以“小端”顺序访问 4 个字节（32 位）。因此在系统端会产生以下属性。

图 2 -9：S7-1500 控制器的数据访问

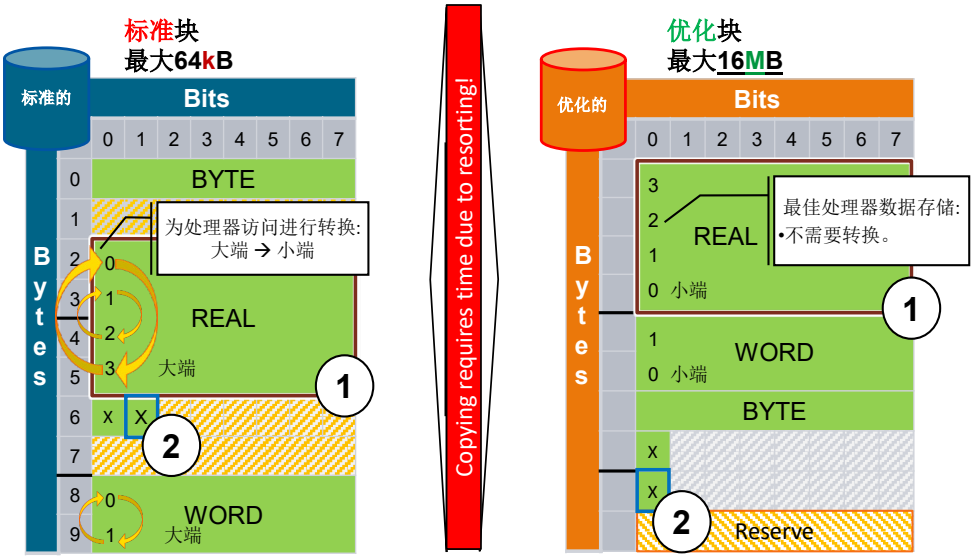


表 2 -10：S7-1500 控制器的数据访问

	标准块	优化块
1.	如果发生不利偏移，控制器需要 2×16 位访问才能读取 4 个字节的值（例如 REAL 值）。 此外，必须翻转字节。	控制器存储变量的访问是优化的。访问是 32 位(REAL)。 不需要翻转字节。
2.	每次访问都会读取整个字节和屏蔽每个位。 整个字节被阻止进行任何其他访问。	每个位都分配一个字节。 控制器在访问时不必屏蔽字节。
3.	最大块大小为 64kB。	最大块大小可达 16MB。

推荐

- 一般来说，只使用优化块。
  - 您不需要绝对寻址，并且始终可以使用符号数据（与对象相关）进行寻址。也可以使用符号数据进行间接寻址（参见第 3.6.2 章 ARRAY 数据类型和间接字段访问）。
  - 在控制器中处理优化块比标准块快得多。
- 避免在优化和非优化块之间复制/赋值数据。源格式和目标格式之间的数据转换需要很长的处理时间。

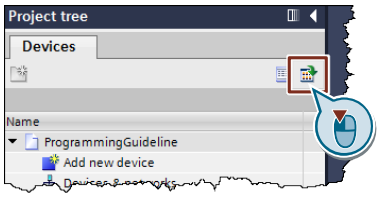
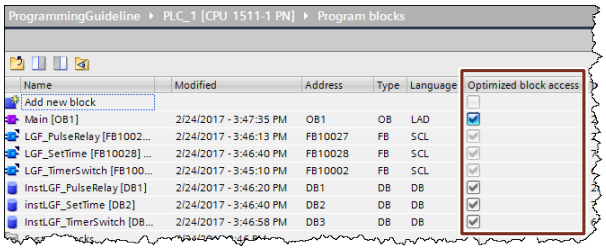
示例：设置优化块访问

默认情况下，为 S7 -1200/1500 的所有新创建的块启用优化块访问。可为 OB、FB 和全局 DB 设置块访问。对于背景 DB，设置来自各自的 FB。

如果将块从 S7-300/400 控制器移植到 S7-1200/1500，块访问不会自动复位设置。您可以稍后将块访问更改为“优化块访问”。更改块访问后，您必须重新编译程序。如果将 FB 更改为“优化块访问”，其分配的背景数据块将自动更新。

按照说明设置优化块访问。

表 2 -411：设置优化块访问

步骤	操作说明
1.	单击项目树中的“最大化/最小化概览”按钮。 
2.	导航到“程序块”。
3.	在这里，您可以看到程序中的所有块以及它们是否经过优化。在此概览中，可以方便地更改“优化块访问”状态。  注意： 背景数据块（此处为“Function_block_1_DB”）从相关 FB 继承“优化”状态。这就是为什么只能在 FB 上更改“优化”设置的原因。编译项目后，DB 会根据相关联的 FB 获得相应状态。

在 TIA 博途中显示优化和非优化的块

在以下两个图中，可以看出优化和非优化背景 DB 之间的差异。

对于全局 DB，存在相同的差异。

图 2 -12：优化数据块（无偏移地址）

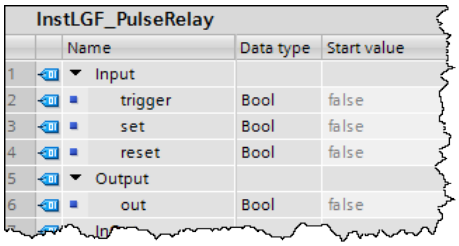


图 2 -13：非优化的数据块（有偏移地址）

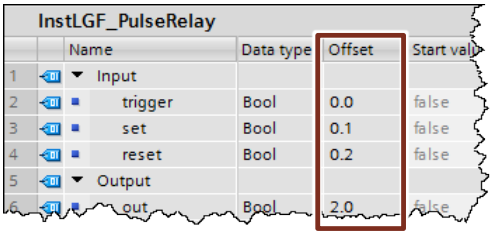


表 2 -14：优化和非优化数据块差异对比

优化数据块	非优化数据块
优化的数据块以符号方式寻址。因此**没有**显示“偏移地址”。	对于非优化的块，“偏移地址”会显示出来，可用于寻址。
在优化块中，您可以使用“保持”单独声明**每个变量。**	在非优化块中，只能使用“保持”声明**所有变量或全部不声明。**

全局 DB 变量的掉电保持性直接在全局 DB 中定义。默认情况下，预设的是掉电不保持。

在函数块（而不是背景 DB）的实例中定义变量的掉电保持性。因此，这些设置对于该 FB 的所有实例都有效。

优化和非优化数据块的访问类型

下表显示了块的所有访问类型。

表 2 -15：访问类型

访问类型	优化块	非优化块
符号化	支持	支持
索引（字段）	支持	支持
片段访问	支持	支持
AT 结构	不支持 （替代方案：片段访问）	支持
直接绝对地址	不支持 （替代方案：带下标的数组）	支持

访问类型	优化块	非优化块
间接绝对地址（指针）	不支持 （替代方案： VARIANT/带下标的数组）	支持
加载而不重新初始化	支持	不支持

注意

更多信息可以在以下条目中找到：

在 STEP 7（TIA 博途）中可以使用哪些访问类型来访问块中的数据值，您应该注意类型之间的差异有哪些？
https://support.industry.siemens.com/cs/ww/en/view/67655611

在使用经过优化访问的 DB 时，应注意 STEP 7（TIA 博途）中指令“READ_DBL”和“WRIT_DBL”的哪些属性？
https://support.industry.siemens.com/cs/ww/en/view/51434747

2.6.4 优化和非优化变量之间的转换

通常建议使用优化的变量。但是，如果在个别情况下希望保留原有的程序，程序中可混合使用优化和非优化的数据存储。

系统知道每个变量的内部存储，无论是结构化的（源自单独定义的数据类型）还是基本的（INT、LREAL、...）。

在不同内存区域的相同类型的两个变量之间赋值，系统会自动转换。这种转换需要对变量结构化，因此应尽可能避免。

2.6.5 优化和非优化访问的块之间的参数传输

当将结构作为输入/输出参数（InOut）传输到被调用块时，它们默认作为引用传输（参见第 3.3.2 章 引用调用）。

但是，如果其中一个块具有“优化访问”属性而另一个块具有“默认访问”属性，则情况并非如此。在这种情况下，所有参数通常作为副本传输（参见第 3.3.1 章按值调用）。

在这种情况下，被调用块始终使用复制的值。在块处理期间，这些值可能会更改，并且在处理块调用后将它们复制回原始操作数。

如果原始操作数被异步进程（例如，被 HMI 或中断 OB 访问）更改，这可能会出问题。如果在块处理之后将复制的值复制回原始操作数，则原始操作数上异步执行的更改将被覆盖。

注意

更多信息可以在以下条目中找到：

为什么 HMI 系统或 Web 服务器的数据有时会在 S7-1500 中被覆盖？
https://support.industry.siemens.com/cs/ww/en/view/109478253

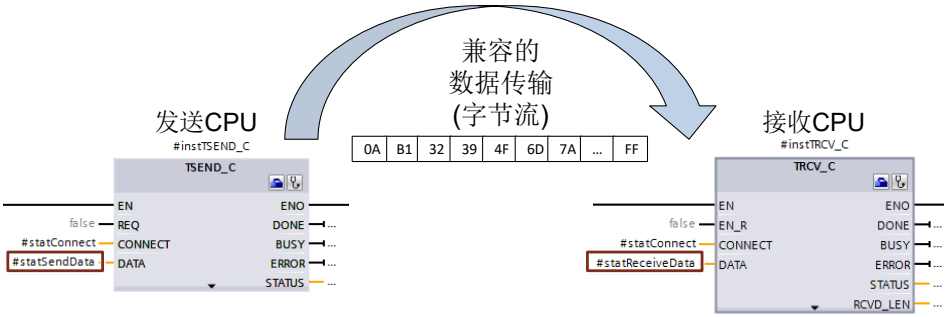
推荐

- 始终为相互通信的两个块设置相同的访问类型。

2.6.6 使用优化数据通信

接口（CPU、CM）以排列方式传输数据（无论是优化还是非优化）。

图 2 -16：CPU-CPU 通信



发送数据可以是:
- 优化的
- 非优化的
- 变量（任意类型）
- 缓存（字节数组）

接收数据可以是:
- 优化的
- 非优化的
- 变量（任意类型）
- 缓存（字节数组）

示例

- 将具有 PLC 数据类型（数据记录）的变量传递给 CPU。
- 在发送 CPU 中，变量作为实际参数与通信块(TSEND_C)互连。
- 在接收 CPU 中，接收数据赋值给相同类型的变量。
- 在这种情况下，可以直接继续对接收到的数据进行符号化操作。

注意 任何变量或数据块都可以用作数据记录（源自 PLC 数据类型）。

注意 也可以以不同的方式定义发送和接收数据：

发送数据		接收数据
优化	-->	非优化
非优化	-->	优化

控制器自动确保数据传输和存储是正确的。

2.7 块属性

2.7.1 块大小

对于 S7-1200/1500 控制器，主存储器中块的最大尺寸明显增大。

表 2 -17：块大小

最大大小和数量（不考虑内存大小）		S7-300/400	S7-1200	S7-1500
DB	最大尺寸	64 KB	64 KB	64 kB 16 MB（优化的 CPU1518）
	最大数量	16.000	65.535	65.535
FC/FB	最大尺寸	64 KB	64 KB	512 KB
	最大数量	7.999	65.535	65.535
FC / FB / DB	最大数量	4.096 (CPU319) 6.000 (CPU412)	1.024	10.000 (CPU1518)

推荐

- 将 S7-1500 控制器的 DB 用作大数据量的数据容器。
- 您可以使用 S7-1500 控制器将>64 kB 的数据量存储在优化的 DB 中（最大大小 16MB）。

2.7.2 组织块(OB)数量

使用 OB 可以创建用户程序层次结构。有不同的 OB 可供使用。

表 2 -18：组织块数量

组织块类型	S7-1200	S7-1500	好处
循环和启动 OB	100	100	用户程序的模块化
硬件中断	50	50	可为每个可能的事件单独创建 OB
延时中断	4 *	20	用户程序的模块化
循环中断		20	用户程序的模块化
时钟中断	无	20	用户程序的模块化

* 从固件 V4 开始，可以有 4 个延时中断和 4 个循环中断。

推荐

- 使用 OB 以分层次构建用户程序。
- 有关使用 OB 的更多建议，请参见第 3.2.1 章组织块(OB)。

2.7.3 块接口–隐藏块参数（V14 或更高版本）

调用块时，块参数可以有针对性的显示或隐藏。在这里，有三个选项可以为每个形式参数单独配置。

- “显示”
- “隐藏”
- “如果未分配参数则隐藏”

优点

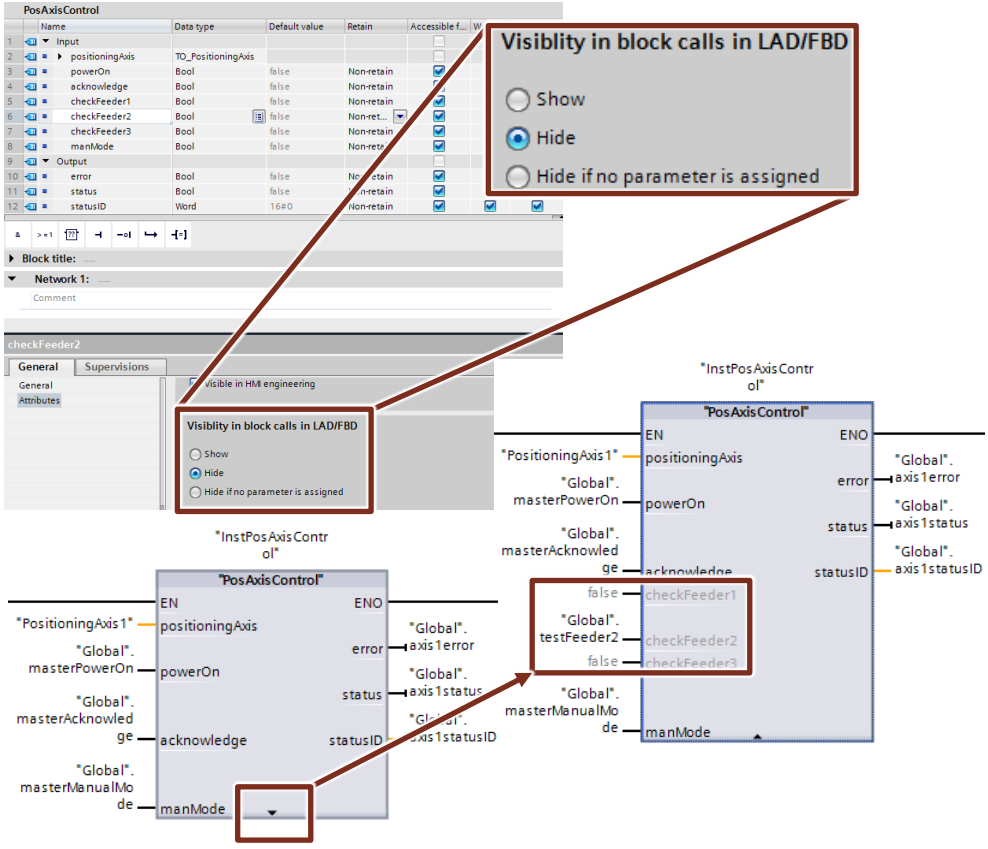
- 更好地概览具有许多可选参数的块

特性

- 可用于：
  - FC, FB
  - In, Out, InOut

示例

图 2 -19：隐藏块参数



2.8 S7-1200/1500 的新数据类型

S7-1200/1500 控制器支持新的数据类型，使编程更加方便。使用新的 64 位数据类型，可以使用更大更精确的值。

注意 更多信息可以在以下条目中找到：

在 STEP 7（TIA 博途）中，如何转换 S7-1200/1500 的数据类型？
https://support.industry.siemens.com/cs/ww/en/view/48711306

2.8.1 基本数据类型

表 2 -20：整数数据类型

类型	大小	数值范围
USint	8 位	0 .. 255
SInt	8 位	-128 .. 127
UInt	16 位	0 .. 65535
UDInt	32 位	0 .. 4.3 Mio(10^9)
ULInt*	64 位	0 .. 18.4 Trio (10^{18})
LInt*	64 位	-9.2 Trio .. 9.2 Trio
LWord	64 位	16#0000 0000 0000 0000 to 16# FFFF FFFF FFFF FFFF

*仅适用于 S7-1500

表 2 -21：浮点数据类型

类型	大小	数值范围
Real	32 位（1 位前缀，8 位指数，23 位尾数），精度为小数点后 7 位	-3.40e+38 .. 3.40e+38
LReal	64 位（1 位前缀，11 位指数，52 位尾数），精度为小数点后 15 位精度	-1.79e+308 .. 1.79e+308

注意 更多信息可以在以下条目中找到：

为什么在 STEP 7（TIA 博途）中，在 SCL 中 DInt 的加法结果显示不正确？

https://support.industry.siemens.com/cs/ww/en/view/98278626

2.8.2 数据类型 Date_Time_Long

表 2 -22：DTL2）的结构

Year	Month	Day	Weekday	Hour	Minute	Second	Nanosecond

DTL 总是读取当前系统时间。通过符号名称访问各个值（例如，`My_Timestamp.Hour`）

优点

- 所有子区域（例如，Year、Month、. . . ）都可以用符号方式寻址。

推荐

使用新的数据类型 DTL 而不是 LDT 并以符号方式对其进行寻址（例如 `My_Timestamp.Hour`）。

注意

更多信息可以在以下条目中找到：

在 STEP 7(TIA 博途)中，如何输入、读取和编辑 S7-300/S7-400/S7-1200/S7-1500 CPU 模块的日期和时间？
https://support.industry.siemens.com/cs/ww/en/view/43566349

STEP 7 V5.5 和 TIA 博途中有哪些函数可用于处理数据类型 DT 和 DTL？
https://support.industry.siemens.com/cs/ww/en/view/63900229

2.8.3 其他时间数据类型

表 2 -23：时间数据类型（仅限 S7-1500）

类型	大小	数值范围
LTime	64 位	LT#-106751d23h47m16s854ms775us808ns 至 LT#+106751d23h47m16s854ms775us807ns
LTIME_OF_DAY	64 位	LTOD#00:00:00.000000000 至 LTOD#23:59:59.999999999

2.8.4 Unicode 数据类型

借助数据类型 WCHAR 和 WSTRING 可以处理 Unicode 字符。

表 2 -24：Unicode 数据类型

类型	大小	数值范围
WCHAR	2 字节	-
WSTRING	(4 + 2*n) 字节	预设值： 0 .. 254 个字符 最大值：0 .. 16382

n = 字符串长度

特性

- 处理例如拉丁文、中文或其他语言的字符。
- 换行符、换页符、制表符、空格
- 特殊字符：美元符号、引号

示例

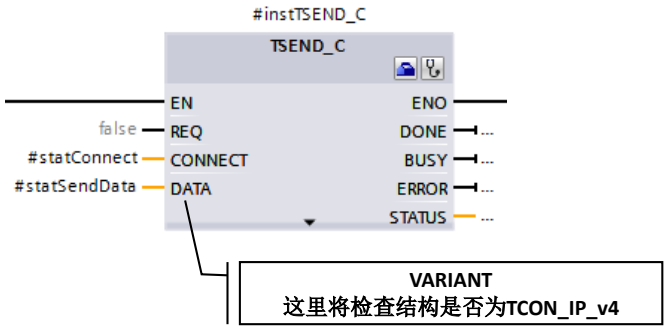
- `WCHAR#'a'`
- `WSTRING#'Hello World!'`

2.8.5 数据类型 VARIANT（S7-1500 和 S7-1200 的 V4.1 以上版本）

VARIANT 类型的参数是一个指针，可以指向不同数据类型的变量。与 ANY 指针相比，VARIANT 是一个带有类型检测的指针。这意味着目标结构和源结构在运行时被检查，必须是相同的。

例如，VARIANT 用于通信块(TSEND_C)作为输入。

图 2 -25：数据类型 VARIANT 作为指令 TSEND_C 的输入参数



优点

- 集成的类型测试可防止错误访问。
- 通过 VARIANT 变量符号寻址，代码更易阅读。
- 代码效率更高，处理时间更短。
- VARIANT 指针显然比 ANY 指针更直观。
- 在系统功能的帮助下，可以直接使用正确类型的 VARIANT 变量。
- 可以灵活高效地传输不同结构的变量。

特性

比较 ANY 和 VARIANT，可以看到以下属性。

表 2 -26：ANY 和 VARIANT 的比较

ANY	VARIANT
使用定义的结构需要 10 个字节的内存	不需要用户的主存储器
通过分配数据区或填充 ANY 结构进行初始化	通过分配数据区或系统指令进行初始化
无类型化的 - 无法识别互连结构的类型	类型化的 - 互连类型可被识别，对于数组，长度也可以确定
部分类型化的 - 对于数组，长度也可以确定	VARIANT 可以通过系统指令进行评估和创建

推荐

- 在必须使用 ANY 指针之前检查一下。在许多情况下，不再需要指针（见下表）。
- 当数据类型仅在程序运行中确定时，仅将数据类型 VARIANT 用于间接寻址。
  - 使用数据类型 VARIANT 作为 InOut 形式参数来创建独立于实际参数数据类型的通用块（参见本章中的示例）。
  - 使用 VARIANT 数据类型而不是 ANY 指针。由于集成的类型测试，可以尽早发现错误。由于符号寻址，程序代码可以很容易地理解。
  - 使用 VARIANT 指令，例如，进行类型识别（参见以下示例和第 2.9.2 章 VARIANT 指令）
- 使用数组的索引而不是通过 ANY 寻址数组元素（参见第 3.6.2 章 ARRAY 数据类型和间接字段访问）。

表 2 -27：比较 ANY 指针和简化

ANY 指针有什么用？		使用 S7-1200/1500 进行简化
可以处理不同数据类型的程序函数	→	使用 VARIANT 指针作为块的 InOut 参数的函数（参见以下示例）
数组的处理 • 例如，读取、初始化、复制相同类型的元素	→	默认的数组功能 • 使用 #myArray[#index] 读写（参见第 3.6.2 章 ARRAY 数据类型和间接字段访问） • 使用 MOVE_BLK 进行复制（参见第 2.9.1 节 MOVE 指令）
• 通过绝对寻址传输结构和高效处理 例如，通过指向函数的 ANY 指针传输用户定义的结构	→	将结构作为 InOut 参数传输 • 见第 3.3.2 章引用调用

注意 如果要复制非结构化 VARIANT 变量的值，您还可以使用 VariantGet 代替 MOVE_BLK_VARIANT（第 2.9.2 章 VARIANT 指令）。

示例

使用数据类型 VARIANT 可以识别用户程序中的数据类型并做出相应的响应。 FC “MoveVariant”的以下代码显示了一种可能的编程方式。

- InOut 形式参数“InVar” （数据类型 VARIANT）用于显示独立于数据类型的变量。
- “Type_Of”指令检测实参的数据类型
- 根据数据类型，使用“MOVE_BLK_VARIANT”指令将变量值复制到不同的输出形式参数。
- 如果未检测到实际参数的数据类型，模块将输出错误代码。

图 2 -28：FC“MoveVariant”的形式参数

MoveVariant

		Name	Data type	Default value
1		Input		
2		Output		
3		outInteger	Int	
4		outReal	Real	
5		outTypeCustom	"typeCustom"	
6		InOut		
7		inOutVariant	Variant	
8		Temp		
9		Constant		
10		Return		

```
CASE TypeOf(#inOutVariant) OF // 检查数据类型

   Int:  // 传输整数
     #MoveVariant := MOVE_BLK_VARIANT(SRC := #inOutVariant,
                                      COUNT := 1,
                                      SRC_INDEX := 0,
                                      DEST_INDEX := 0,
                                      DEST => #outInteger);

   Real:  // 传输实数
     #MoveVariant := MOVE_BLK_VARIANT(SRC := #inOutVariant,
                                      COUNT := 1,
                                      SRC_INDEX := 0,
                                      DEST_INDEX := 0,
                                      DEST => #outReal);

   typeCustom:  // 传输 outTypeCustom
      #MoveVariant := MOVE_BLK_VARIANT(SRC := #inOutVariant,
                                      COUNT := 1,
                                      SRC_INDEX := 0,
                                      DEST_INDEX := 0,
                                      DEST => #outTypeCustom);

   ELSE  //错误，没有符合条件的数据类型
      #MoveVariant := WORD_TO_INT(#NO_CORRECT_DATA_TYPE);
      // 80B4: MOVE_BLK_VARIANT 的错误代码：数据类型不一致
END_CASE;
```

2.9 指令

TIA 博途为程序员提供了现成的指令（位逻辑、计时器、计数器、比较……）。

注意

更多函数可在以下条目中下载：

用于 STEP 7(TIA 博途)和 S7-1200/S7-1500 的(LGFP)通用函数库
https://support.industry.siemens.com/cs/ww/en/view/109479728

2.9.1 MOVE 指令

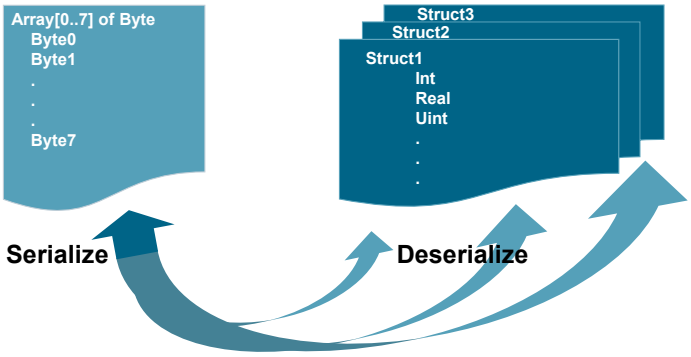
在 STEP 7(TIA 博途）中，可以使用以下 MOVE 指令。 MOVE_BLK_VARIANT 指令是 S7-1200/1500 的新指令。

表 2 -29：MOVE 指令

指令	用途	特性
MOVE	复制值	• 将输入 IN 的参数内容复制到输出 OUT 的参数。 • 输入和输出的参数必须是相同的数据类型。 • 参数也可以是结构体变量（PLC 数据类型）。 • 复制完整的数组和结构。
MOVE_BLK	复制数组	• 将数组的内容复制到另一个数组。 • 源数组和目标数组必须是相同的数据类型。 • 复制完整的数组和结构。 • 也可以复制结构的几个数组元素。此外，可以指定元素的起点和数量。
UMOVE_BLK	无中断复制数组	• 持续地复制数组的内容，而不会有被 OB 中断复制过程的风险。 • 源数组和目标数组必须是相同的数据类型。
MOVE_BLK_VARIANT （S7-1500 和 S7-1200 FW4.1 或更高版本）	复制数组	• 复制一个或多个结构体变量（PLC 数据类型） • 在运行时识别数据类型 • 提供详细的错误信息 • 除了基本数据类型和结构体数据类型外，还支持 PLC 数据类型、数组和数组 DB。
Serialize （S7-1500 和 S7-1200 FW4.1 或更高版本）	将结构体数据转换为字节数组	• 多个数据记录可以组合成一个字节数组，例如，作为消息帧发送到其他设备。 • 输入和输出参数可以使用数据类型 Variant 传输。
Deserialize	将一个字节数组转换为一个或多个结构体	• 应用案例智能设备：智能设备在输入区域接收到多个数据记录，这些数据记录被复制到不同的结构体中。

指令	用途	特性
（S7-1500 和 S7-1200 FW4.1 或更高版本）		• 多个数据记录可以组合成一个单字节数组。Deserialize 可以将这些记录复制到不同的结构体中。

图 2 -30：Serialize 和 Deserialize（S7-1500 和 S7-1200 FW4.1 或更高版本）



特性

“Serialize”、“Deserialize”、“CMP”（比较）和“MOVE：复制值”等指令可以处理非常大且复杂的结构体变量。在此过程中，CPU 在运行时分析变量结构。处理时间取决于要处理的变量结构的以下属性：

- 结构体的复杂性
- 不使用 PLC 数据类型的结构体数量
- 字节数组可以保存在优化块（V14 或更高版本）中。

推荐

- 借助 PLC 数据类型而不是“STRUCT”声明结构体
- 减少使用的结构体数量：
  - 例如，避免对非常相似的结构体进行多次声明。将它们总结为一个单一的结构体。
  - 当结构体的许多元素具有相同的数据类型时，如果可能，请使用此数据类型数组。

- 一般需要区分 MOVE、MOVE_BLK 和 MOVE_BLK_VARIANT
  - 使用 MOVE 指令复制完整的结构体。
  - 使用 MOVE_BLK 指令复制已知数据类型的数组的一部分。
  - 如果您希望复制具有仅在程序运行时才知道的数据类型的数组部分，请仅使用 MOVE_BLK_VARIANT 指令。

注意 UMOVE_BLK：复制过程不能被操作系统的其他活动中断。因此，在处理“无中断复制数组”指令时，CPU 的报警反应时间可能会增加。

有关 MOVE 指令的完整说明，请参阅 TIA 博途在线帮助。

注意 更多信息可以在以下条目中找到：

如何在 STEP 7(TIA 博途)中复制存储区和结构体数据？
https://support.industry.siemens.com/cs/ww/en/view/42603881

2.9.2 VARIANT 指令（S7-1500 和 S7-1200 的 V4.1 以上版本）

表 2 -31：VARIANT 指令

指令	用途	特性
VARIANT 指令		
VariantGet	读取值	该指令使您能够读取指向 VARIANT 变量的值。
VariantPut	写入值	该指令使您能够写入指向 VARIANT 变量的值。
枚举		
CountOfElements	元素计数	使用此指令，您可以获得指向 VARIANT 变量的 ARRAY 元素的数量。
比较指令		
TypeOf()（仅限 SCL）	确定数据类型	使用此指令获得指向 VARIANT 变量的数据类型。
TypeOfElements()（仅限 SCL）	确定数组数据类型	使用此指令获得指向 VARIANT 变量的 ARRAY 的元素数据类型。

指令	用途	特性
比较指令		
VARIANT_TO_DB_ANY（仅限 SCL）	确定数据块号	该指令查询基于 PLC 数据类型、系统数据类型创建的数据块或数组 DB、背景数据块的数据块号。
DB_ANY_TO_VARIANT（仅限 SCL）	从 VARIANT 变量的数据块创建。	该指令基于 PLC 数据类型、系统数据类型创建的数据块或数组 DB、背景数据块创建 VARIANT 变量。

注意 有关 VARIANT 的更多说明，请参阅 TIA 博途的在线帮助。

特性

由于其复杂的算法，VARIANT 指令需要比直接指令更长的处理时间。

推荐

- 如果可能，不要在循环（FOR、WHILE…）中使用 VARIANT 指令，以防止不必要地增加循环时间。
- 不要通过使用循环元素来复制数组，而是直接赋值完整的数组。

2.9.3 RUNTIME 指令

“RUNTIME”指令测量整个程序、单个块或命令序列的运行时间。您可以在 LAD、FBD、SCL 和 STL（仅限 S7-1500）中调用此指令。

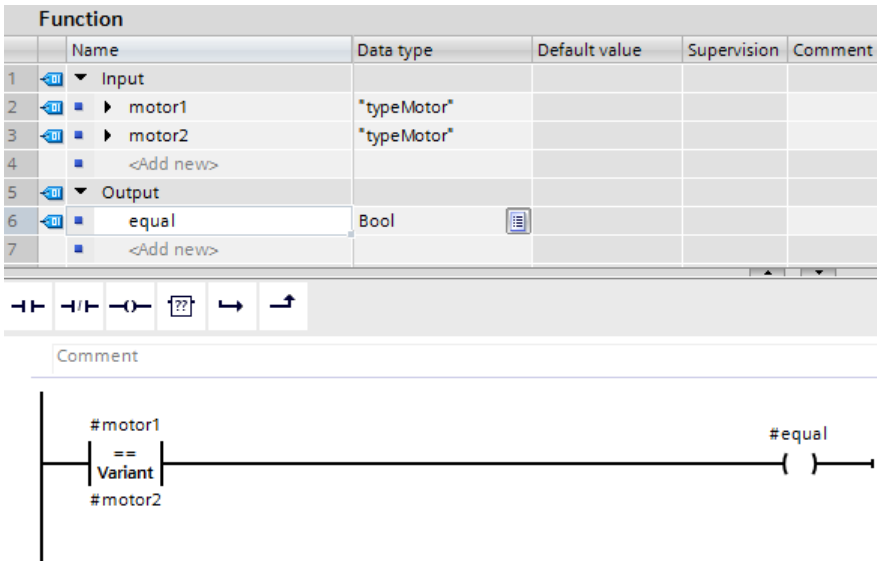
注意 更多信息可以在以下条目中找到：

使用 S7-1200/S7-1500，如何测量组织块的总循环时间？
https://support.industry.siemens.com/cs/ww/en/view/87668055

2.9.4 PLC 数据类型的变量比较（V14 或更高版本）

可以检查相同 PLC 数据类型的两个变量值是否相同。

图 2 -32：LAD 中 PLC 数据类型的变量比较

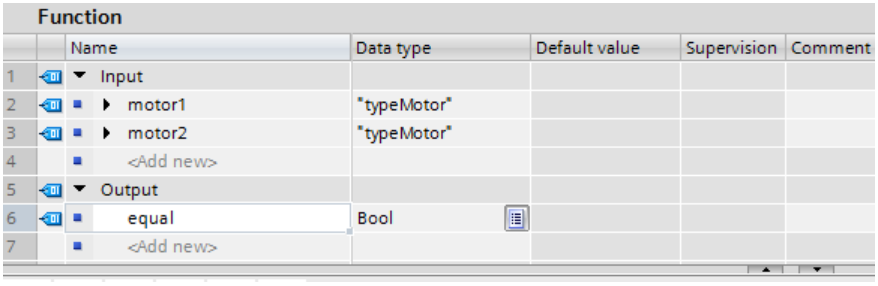


优点

- 使用结构体变量进行符号编程
- 最佳性能的比较
- 可以在 LAB、FBD、STL 中进行比较。
- 在 STL 指令中可以直接进行比较。

示例

图 2 -33：SCL 指令中 PLC 数据类型的变量比较




```
IF #motor1 = #motor2 THEN
   // IF 语句部分
   ;
END_IF;
```

2.9.5 多重赋值（V14 或更高版本）

优点

多重赋值可以优化多个变量的编程（例如，用于初始化）。

示例

```
# statFillLevel : = #statTemperature := #tempTemperature := 0.0;
```

2.10 符号和注释

2.10.1 编程编辑器

优点

通过在程序中使用符号名称和注释，您可以使代码易于理解和阅读。
完整的符号是与程序代码一起下载到控制器的，因此即使在没有离线项目可用的情况下也可以快速维护。

推荐

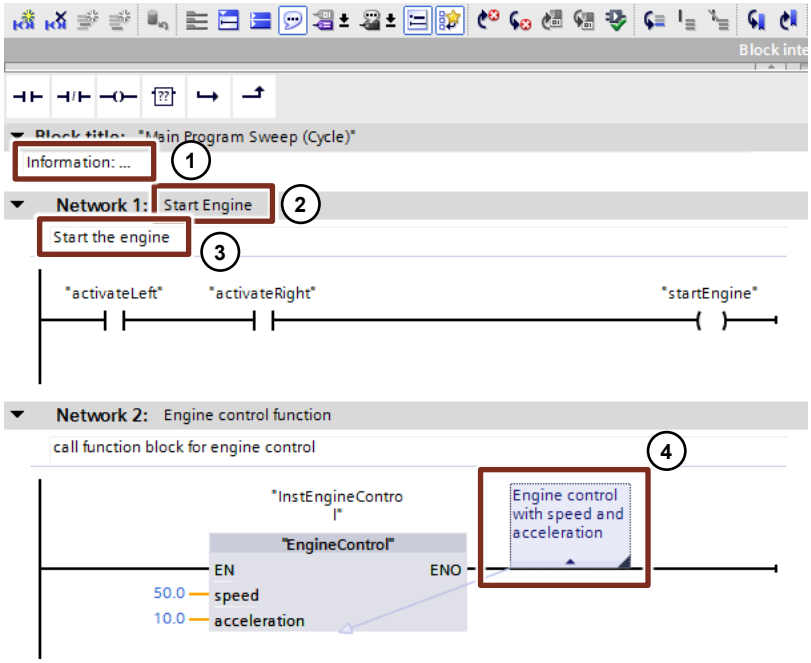
- 在程序中使用注释以提高可读性。即使网络已折叠，网络标题注释也是可见的。
- 以便于同事也可以立即理解程序的方式设计程序代码。

在以下示例中，您可以看到用于在编辑器中注释程序的广泛选项。

示例

在下图中，您可以看到 LAD 编辑器中的注释选项（FBD 中的功能相同）。

图 2 -34：在用户程序中注释 (LAD)



以下注释是可能的：

1. 块注释
2. 网络标题注释
3. 网络注释
4. 注释指令、块和函数（打开、关闭等）

在编程语言 SCL 和 STL 中，可以在每一行中用//进行注释。

示例

```
statFillingLevel : = statRadius * statRadius * PI * statHight ;
//计算中型水箱的液位
```

注意

有关详细信息，请参阅以下条目：

在 STEP 7(TIA 博途）中，为什么在块编辑器中打开项目后不再显示文本、标题和注释？
https://support.industry.siemens.com/cs/ww/en/view/41995518

2.10.2 监控表中的注释行

优点

- 为了更好的结构，可以在监控表中创建注释行。

推荐

- 始终使用注释行并细分您的监控表。
- 还请注释各个变量。

示例

图 2 -35：带有注释行的监控表

	i	Name	Address
1		// Building 1 floor 3 room 21	
2		"Building".fanSpeed1	
3		"Building".temperature1	
4		"Building".light1	
5		// Building 2 floor 2 room 48	
6		"Building".fanSpeed2	
7		"Building".temperature2	
8		"Building".light2	
9		// Building 4 floor 4 room 77	
10		"Building".fanSpeed3	
11		"Building".temperature3	
12		"Building".light3	

2.11 系统常量

对于 S7-300/400 控制器，硬件和软件组件的识别由逻辑地址或诊断地址执行。

对于 S7-1200/1500，识别是通过系统常量进行的。S7-1200/1500 控制器的所有硬件和软件组件（例如，接口、模块、OB. . . ）都有自己的系统常量。系统常量是在中央和分布式 I/O 的设备组态的设置期间自动创建的。

2.11 操作系统与用户程序

优点

- 可以通过模块名称而不是硬件标识来寻址。

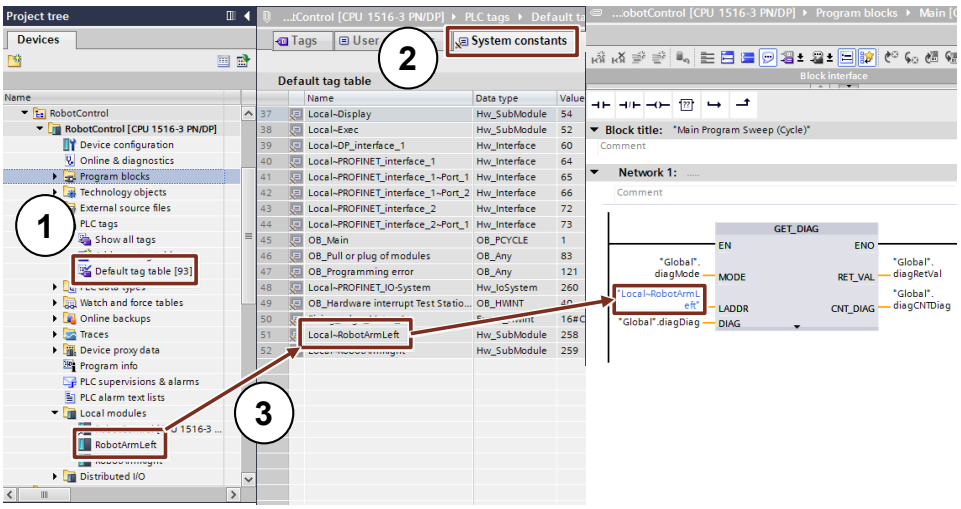
推荐

- 分配与函数相关的模块名称，以便在编程过程中轻松识别模块。

示例

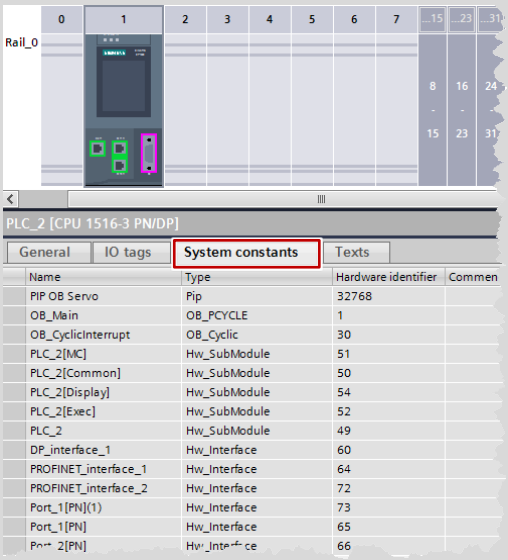
在以下示例中，可以看到系统常量是如何在用户程序中使用的。

图 2 -36：用户程序中的“系统常量”



1. 控制器的系统常量可在“PLC 变量 - 默认变量表”文件夹中找到。
2. 系统常量位于“默认变量表”的单独列表中。
3. 在此示例中，为 DI 模块分配了符号名称“RobotArmLeft”。
   您也可以在系统常量表中找到该名称下的模块。
   在用户程序中，“RobotArmLeft”与“GET_DIAG”诊断块互连。

注意 打开“设备组态”可以快速找到每个设备的系统常量。



Name	Type	Hardware identifier	Commen
PIP OB Servo	Pip	32768	
OB_Main	OB_PCYCLE	1	
OB_CyclicInterrupt	OB_Cyclic	30	
PLC_2[MC]	Hw_SubModule	51	
PLC_2[Common]	Hw_SubModule	50	
PLC_2[Display]	Hw_SubModule	54	
PLC_2[Exec]	Hw_SubModule	52	
PLC_2	Hw_SubModule	49	
DP_interface_1	Hw_Interface	60	
PROFINET_interface_1	Hw_Interface	64	
PROFINET_interface_2	Hw_Interface	72	
Port_1[PN](1)	Hw_Interface	73	
Port_1[PN]	Hw_Interface	65	
Port_2[PN]	Hw_Interface	66	

注意 更多信息可以在以下条目中找到：

STEP 7 (TIA 博途) 中的系统常量对 S7-1200/1500 有什么意义？
https://support.industry.siemens.com/cs/ww/en/view/78782835

2.12 用户常量

常量值可以通过用户常量的帮助保存。通常，控制器中有 OB、FC 和 FB 的局部常量和整个用户程序的全局常量。

优点

- 用户常量可用于更改全局或本地所有使用位置的常量值。
- 使用用户常量，可以使程序更具可读性。

特性

- 本地用户常量在块接口中定义。
- 全局用户常量在“PLC 变量”中定义。
- 用户程序仅可对用户常量读取访问。
- 对于受专有技术保护的块，用户常量是不可见的。

推荐

- 使用用户常量来提高程序的可读性和集中可变性
  - 错误代码，
  - CASE 指令，
  - 转换系数，
  - 自然常数...

示例

图 2 -37：用于 CASE 指令的本地用户常量

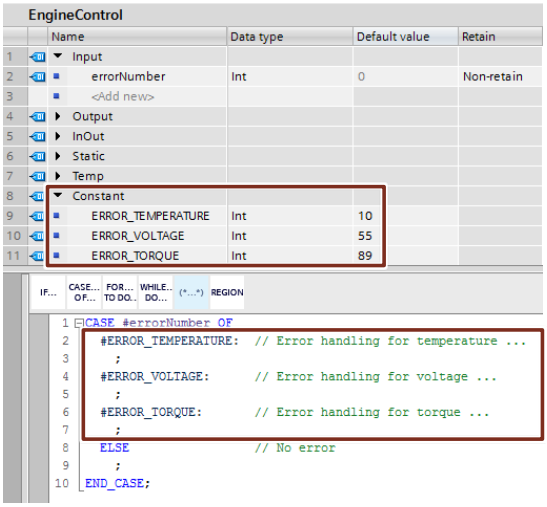
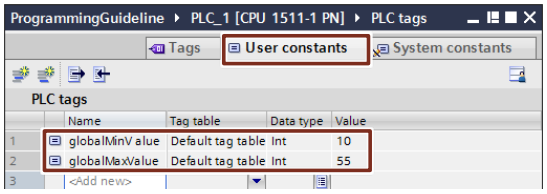


图 2 -38：控制器的全局用户常量



注意

以下常见问题解答中提供了常量的另一个应用案例：

如何在 STEP 7（TIA 博途）中转换变量的单位？
https://support.industry.siemens.com/cs/ww/en/view/61928891

2.13 控制器和 HMI 变量的内部参考 ID

STEP 7、WinCC、Startdrive、Safety 等集成到 TIA 博途工程框架的联合数据库中。用户程序中的所有位置都会自动接受数据更改，无论更改是发生在控制器、触摸屏还是驱动器中。因此不会出现数据不一致的情况。

如果创建变量，TIA 博途会自动创建唯一的参考 ID。您无法查看或设置参考 ID。此过程是内部引用。更改变量（地址）时，参考 ID 保持不变。

在下图中，示意性地显示了对数据的内部引用。

图 2 -39：PLC 和 HMI 的内部参考 ID

PLC1 **HMI1**

PLC 变量名	绝对地址	PLC 内部参考ID	HMI 内部参考ID	HMI变量名	存取方式	与PLC连接
motor1	I0.0	000123	009876	motor1	<符号访问>	PLC1_HMI1
valve2	Q0.3	000138	000578	valve2	<符号访问>	PLC1_HMI1

注意

ID 在以下情况下会改变...

- 重命名变量。
- 改变类型。
- 删除变量。

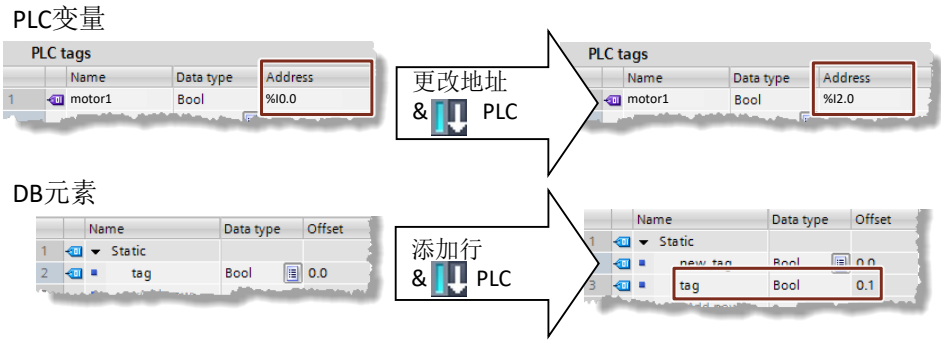
优点

- 您可以在不改变内部关系的情况下重新连接变量。控制器、HMI 和驱动器之间的通信也保持不变。
- 符号名称的长度对控制器和 HMI 之间的通讯负载没有影响。

特性

如果更改 PLC 变量的地址，则只需重新加载控制器，因为系统还使用参考 ID 在内部对系统进行寻址。无需重新加载 HMI 设备（参见图 2 -24：更改地址或添加行）。

图 2 -40：更改地址或添加行



2.14 发生错误时的 STOP 模式

与 S7-300/400 相比，S7-1200/1500 导致“STOP”模式的条件更少。

由于 TIA 博途中更改了一致性检查，大多数情况下已经可以预先防止 S7-1200/1500 控制器进入“STOP”模式。在 TIA 博途编译时已经检查了程序块的一致性。这种方法使 S7-1200/1500 控制器比其前代产品更能“容错”。

优点

只有三种故障情况会使 S7-1200/1500 控制器进入 STOP 模式。这使得错误管理的编程更加清晰和容易。

特性

表 2 -41：对 S7-1200/1500 错误的响应

	错误	S7-1200	S7-1500
1.	周期监控超时一次	RUN	**STOP** （未配置 OB80 时）
2.	周期监控超时两次	**STOP**	**STOP**
3.	编程错误	RUN	**STOP** （未配置 OB121 时）

错误 OB：

- 当超过控制器的最大循环时间时，操作系统会调用 OB80“时间错误中断”。
- 当程序执行期间发生错误时，操作系统会调用 OB121“编程错误”。

此外，对于每个错误，都会在诊断缓冲区中自动创建一个条目。

注意 对于 S7-1200/1500 控制器，还有其他可编程的错误 OB（诊断错误、模块机架故障等）。

有关 S7-1200/1500 错误响应的更多信息，请参见 TIA 博途在线帮助中的“事件和 OB”。

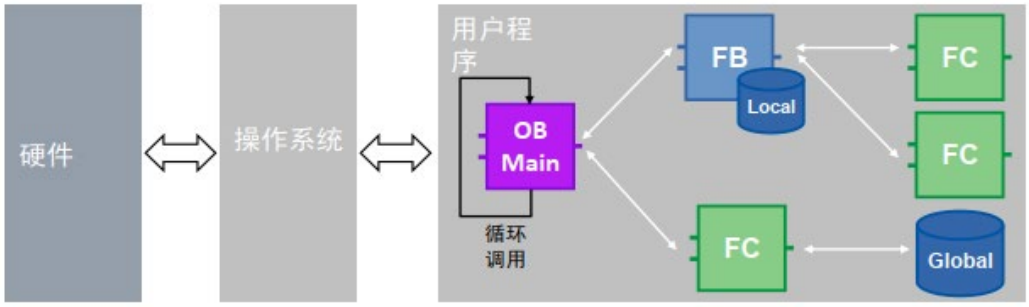
3 通用编程

3.1 操作系统与用户程序

SIMATIC 控制器由操作系统和用户程序组成。

- 操作系统管理所有未与特定控制任务连接的函数和序列(例如：处理重启、更新过程映像、调用用户程序、错误处理、内存管理等)。操作系统是控制器不可分割的一部分。
- 用户程序包括处理特定自动化任务所需的所有块。用程序块对用户程序进行编程，并加载到控制器上。

图 3 -1：操作系统与用户程序



对于 SIMATIC 控制器，用户程序总是循环执行。在 STEP 7 中创建控制器后，“Main”循环 OB 已经存在于 “程序块” 文件夹中。该块由控制器处理，并被无限循环调用。

3.2 程序块

在 STEP 7(TIA 博途)中，有所有熟悉的块类型来自之前的 STEP 7 版本：

- 组织块
- 函数块
- 函数
- 数据块

有经验的 STEP 7 用户会马上知道他们的方法，新用户可以很容易地熟悉编程。

优势

- 使用不同的块类型给你的程序一个清晰的结构。
- 基于一个良好的和结构化的程序，你可以得到许多函数单元，它们可以在一个项目和其他项目中多次重复使用。这些函数单元通常只在不同的配置上有所不同(见章节 3.2.9 块的重用性)。
- 你的项目或你的工厂将变得更加透明。也就是说，一个工厂的错误状态可以更容易地被检测、分析和消除。换句话说，工厂的可维护性变得更容易了。对于编程中的错误也是如此。

建议

- 结构化自动化任务。
- 将工厂的整个功能划分为独立的区域，形成子函数单元。将这些函数单元再次划分为更小的单元和函数。直到得到可以多次使用并带有不同参数的函数。
- 指定函数单元之间的接口。为将要由“外部伙伴”交付的函数，定义独特的接口。

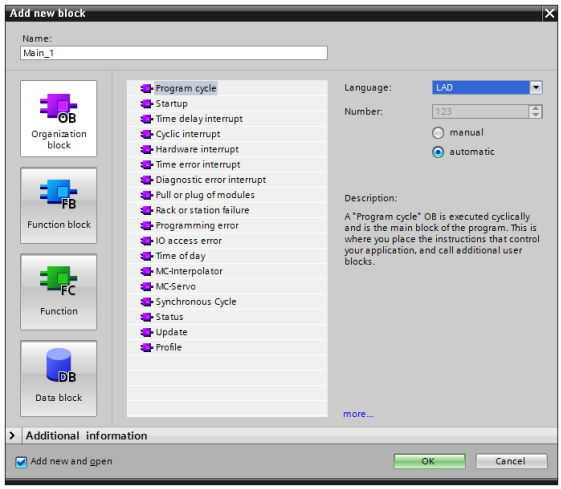
所有组织块、函数块和函数都可以用以下语言编程：

表 3 -2：编程语言

编程语言	S7-1200	S7-1500
梯形图 (LAD)	是	是
函数块图 (FBD)	是	是
结构化控制语言 (SCL)	是	是
Graph	否	是
语句表 (STL)	是	是

3.2.1 组织块(OB)

图 3 -3：“添加新块” 对话框(OB)



OB 是操作系统和用户程序之间的接口。它们由操作系统调用并控制，例如以下程序：

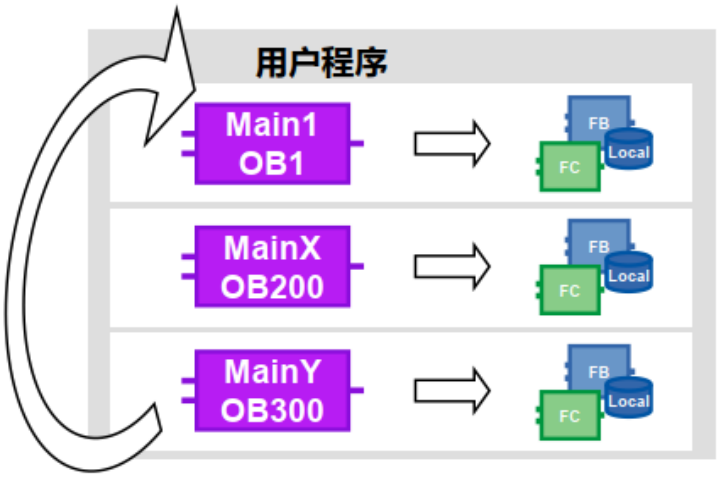
- 控制器的启动行为
- 循环程序处理
- 中断控制程序处理
- 错误处理

根据控制器的不同，可以使用许多不同的 OB 类型。

属性

- OB 由控制器的操作系统调用。
- 可以在一个程序中创建几个主 OB。OB 按 OB 号顺序被处理。

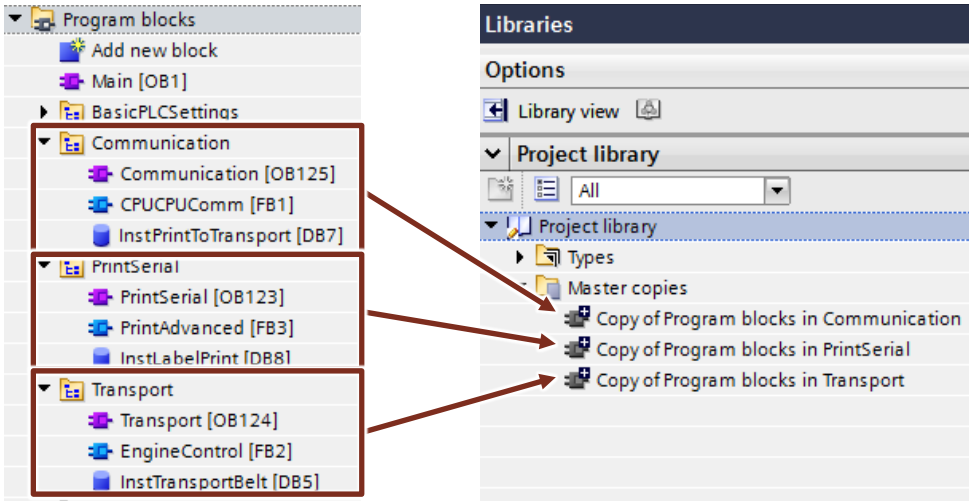
图 3 -4：使用几个主要的 OB



建议

- 将不同的程序部件封装到几个主 OB 中，这些程序部件在控制器之间可能是可替换的。
- 避免不同主 OB 之间的通信。由此可以相互独立地使用它们。如果您在各个主 OB 之间交换数据，请使用全局 DB（见章节 4.2 不使用位储存器而使用全局数据块）。
- 将彼此属于对方的所有程序部分划分到文件夹中，并将它们存储在项目库或全局库中以供重新使用。

图 3 -5：将程序部分按顺序存储在项目库中



更多信息见章节 3.7 库。

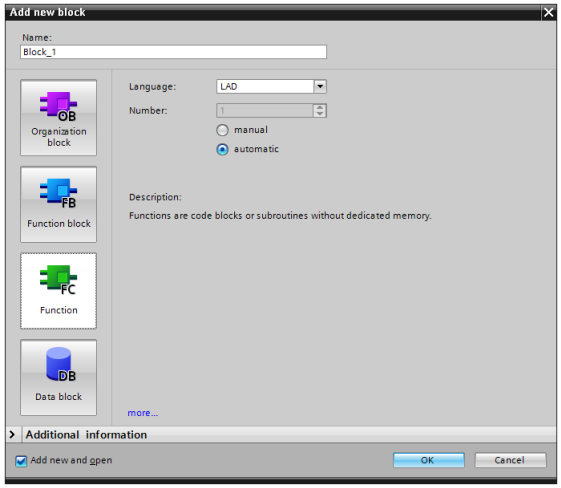
注意

更多资料可参阅下列条目：

哪些组织块可以在 STEP 7 (TIA 博途) 中使用?
https://support.industry.siemens.com/cs/ww/en/view/40654862

3.2.2 函数(FC)

图 3 -6：“添加新块”对话框 (FC)



FC 是没有循环数据存储的块。这就是为什么块的参数值不能保存到下一次调用，并且在调用时必须提供实际参数的原因。

属性

- FC 是没有循环数据存储的块。
- 在非优化块中调用临时变量时，临时变量未被定义。在优化块中，该值始终预置为默认值 (S7-1500 和 S7-1200 固件 V4 及更高版本)。因此，由此产生的特性不是偶然的，而是可复现的特性。
- 为了永久保存 FC 的数据，可以使用全局数据块的功能。
- FC 可以有多个输出。
- 函数值可以直接在 SCL 中的公式被再次使用。

建议

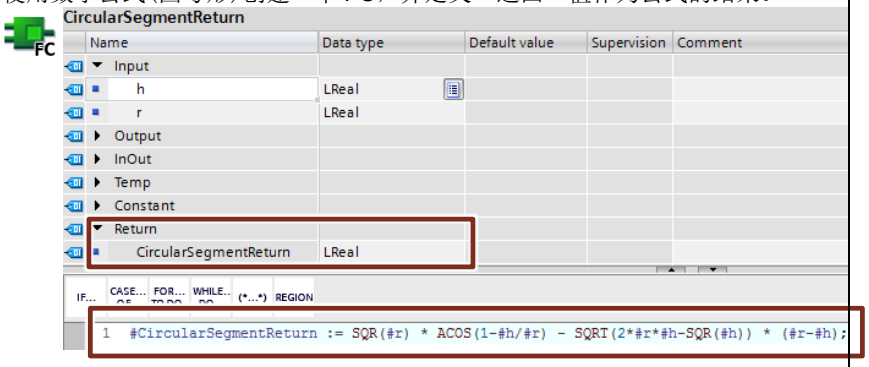
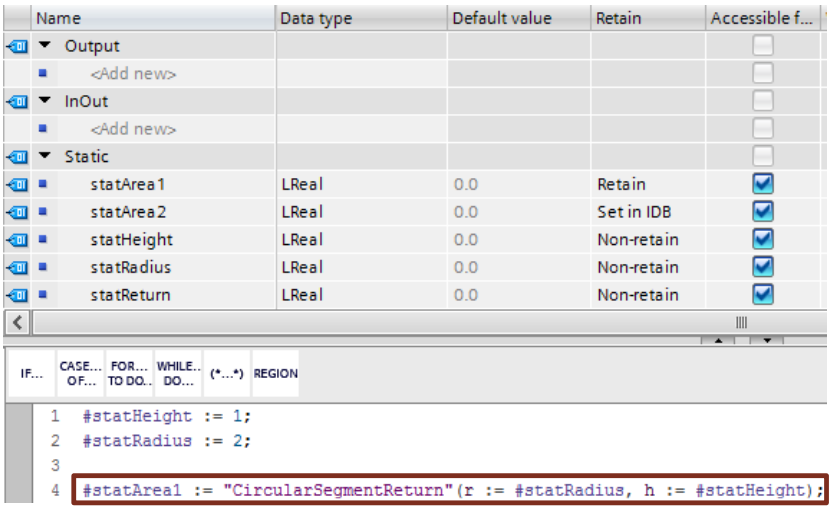
- 对于被多次调用且频繁重复出现的应用程序，可在用户程序的不同位置使用这些函数。
- 使用该选项可以在 SCL 中直接重用函数值。

```
<Operand> := <FC name> (Parameter list);
```

示例

在下面的示例中，在 FC 中编写了一个数学公式。计算结果直接被声明为返回值，函数值可以直接重复使用。

表 3 -7：重复使用函数值

步骤	说明
1.	使用数学公式(园弓形)创建一个 FC，并定义“返回”值作为公式的结果。 
2.	在任意块(SCL)中调用 FC 使用圆段计算。 <Operand> := <FC name> (parameter list); 

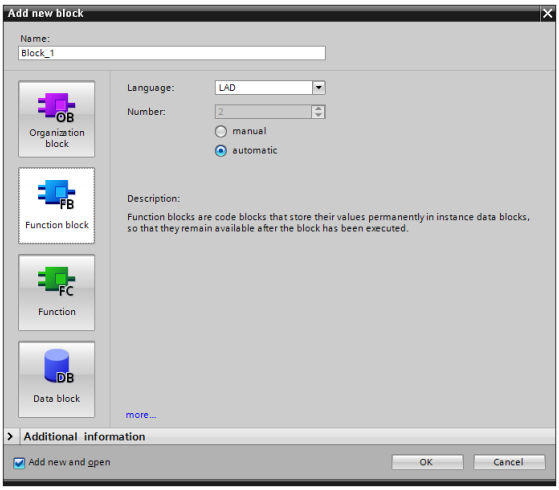
注意

更多资料可参阅下列各项：

对于 S7-1200/S7-1500 CPU 的一个函数，允许在 STEP 7(TIA 博途)中定义的最大参数个数是多少?
https://support.industry.siemens.com/cs/ww/en/view/99412890

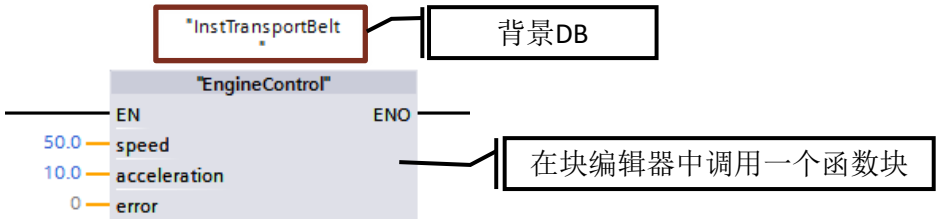
3.2.3 函数块(FB)

图 3 -8：“添加新块” 对话框 (FB)



FB 是具有循环数据存储的块，其中的值是永久存储的。循环数据存储在背景数据块中实现。

图 3 -9：调用一个函数块



属性

- FB 是具有循环数据存储的块。
- 在非优化块中调用临时变量时，未定义临时变量。在优化块中，该值始终预置为默认值(S7-1500 和 S7-1200 固件 V4)。因此，由此产生的行为不是偶然的，而是可复现的。
- 静态变量保存每个周期的值。

建议

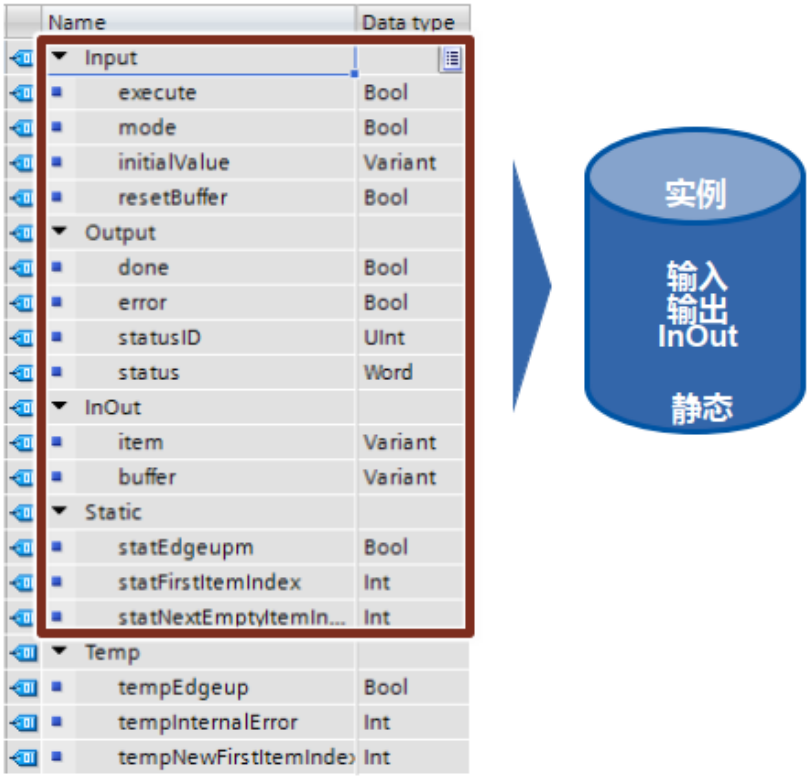
- 使用函数块来创建子程序和构造用户程序。函数块也可以在用户程序的不同位置被多次调用。这使得频繁重复的程序部分的编程更容易。
- 如果函数块在用户程序中多次应用，请使用单独的实例，最好是多重实例。

3.2.4 实例

函数块的调用被称为实例。实例正在使用的数据保存在背景数据块中。

背景数据块总是根据 FB 接口中的规范创建的，因此不能在背景数据块中更改。

图 3 -10：FB 的接口结构



背景数据块由一个包括输入、输出、InOut 和静态接口的永久内存组成。临时变量存储在易失性存储器(L 栈)中。L 栈始终只对当前进程有效。也就是说，临时变量必须在每个周期中初始化。

属性

- 背景数据块总是被分配给 FB。
- 背景数据块不需要在 TIA 博途中手动创建，而是可以在调用 FB 时自动创建。
- 背景数据块的结构在相对应的 FB 中指定，并且只能在那里更改。

建议

- 某种程度在程序中，背景数据块的数据只能由相对应的 FB 更改。这就是如何保证块可以在所有类型的项目中被普遍使用。

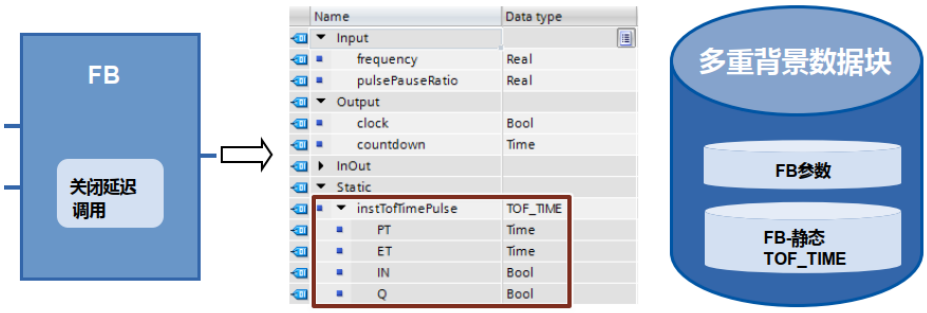
更多信息请参考章节 3.4.1 块接口的数据交换。

3.2.5 多重实例

使用多重实例调用函数块可以将它们的数据存储在被调用函数块的背景数据块中。这意味着，如果在一个函数块中调用另一个函数块，它将其数据保存在更高级别 FB 的背景数据块中。因此，即使在传输调用块时，也要维护被调用块的功能性。

下图显示了使用另一个 FB（“IEC 计时器”）的 FB。所有数据保存在一个多重背景数据块中。因此，可以创建具有独立时间行为的块，例如，时钟生成器。

图 3 -11：多重实例



优势

- 可重用性。
- 可以进行多个调用。
- 更清晰的程序与更少的背景数据块。
- 简单的程序复制。
- 对于编程中的结构化是个好选择。

属性

- 多重实例是背景数据块中的内存区域。

建议

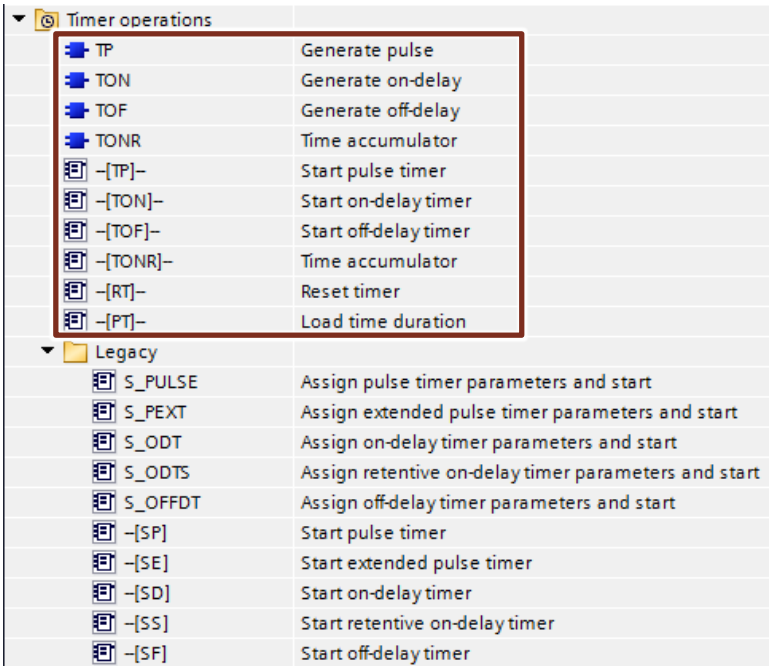
使用多重实例可以…

- 减少背景数据块的数量。
- 创建可重复使用和清晰的用户程序。
- 编写本地函数，例如定时器，计数器，边缘计算。

示例

如果你需要时间和计数器功能，使用“IEC 计时器”块和“IEC 计数器”块，而不是绝对地址的 SIMATIC 计时器。如果可能的话，在这里可以使用多重实例。因此，用户程序中的块数量将保持在较低的水平。

图 3 -12：IEC 计时器的库



注意

更多资料可参阅下列各项：

对于 S7-1500 如何在 STEP 7 (TIA 博途) 中声明计时器和计数器?
https://support.industry.siemens.com/cs/ww/en/view/67585220

3.2.6 作为参数的实例传递 (V14)

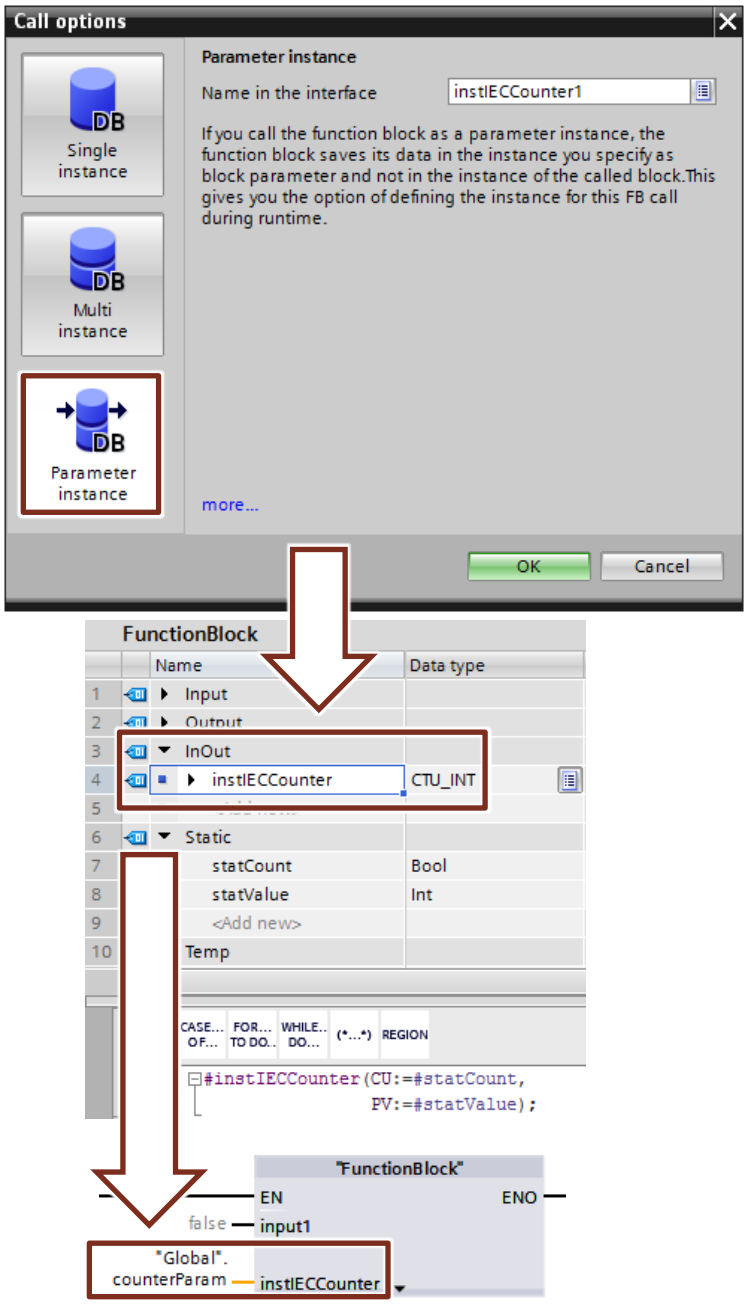
被调用块的实例可以被定义为 InOut 参数。

优势

- 可以创建标准化的函数，其动态实例被传递。
- 只有在调用块的时候才会指定使用哪个实例。

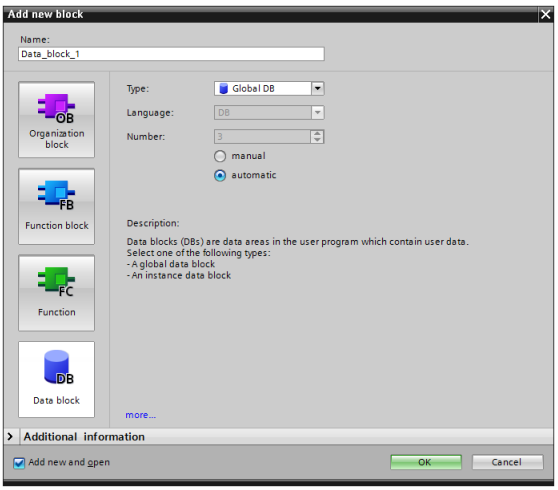
示例

图 3 -13：作为参数的实例传递



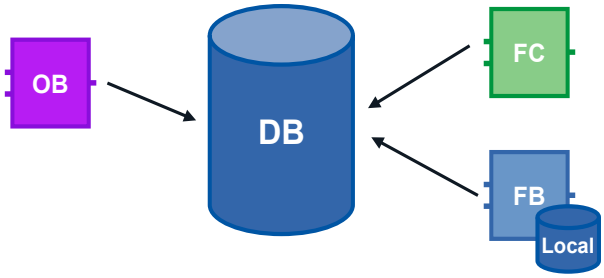
3.2.7 全局数据块(DB)

图 3 -14：“添加新块” 对话框(DB)



变量数据位于整个用户程序可用的数据块中。

图 3 -15：全局数据块作为中央数据存储器



优势

- 结构良好的存储区域。
- 访问速度快。

属性

- 用户程序中的所有块都可以访问全局数据块。
- 全局数据块的结构可以由所有数据类型任意组成。
- 全局数据块可以通过程序编辑器创建，也可以根据之前创建的“用户自定义 PLC 数据类型”创建（见章节 3.6.4 STRUCT 数据类型和 PLC 数据类型）。
- 最多可以定义 256 个结构化变量(ARRAY，STRUCT)。这不适用于从 PLC 数据类型派生的变量。

建议

- 当数据在不同的程序部分或块使用时，使用全局数据块。

注意

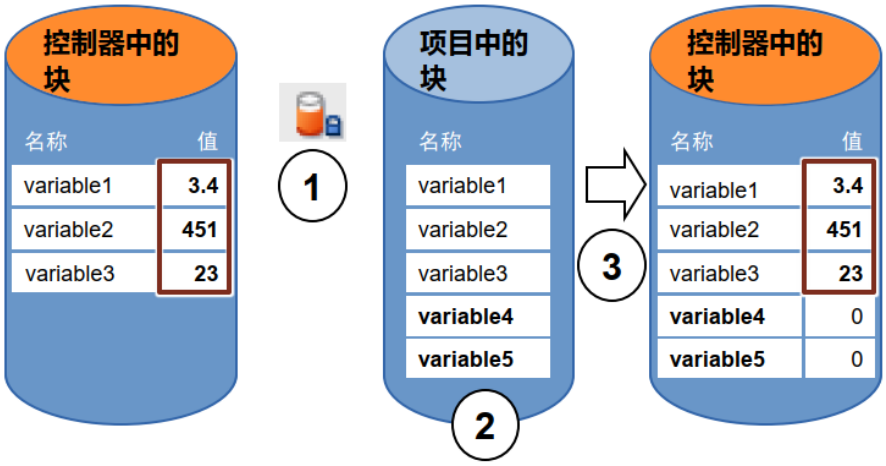
更多资料可参阅下列各项：

STEP 7（TIA 博途）中全局数据块的声明表是如何结构化的?
https://support.industry.siemens.com/cs/ww/en/view/68015630

3.2.8 下载但不重新初始化

为了改变已经在控制器中运行的用户程序，S7-1200（固件 V4.0）和 S7-1500 控制器提供了在运行过程中扩展优化函数块或数据块接口的选项。您可以在不将控制器设置为 STOP 的情况下加载更改的块，也不会影响已经加载的变量的实际值。

图 3 -16：下载但不重新初始化



在控制器处于 RUN 模式时，执行以下步骤。

1. 启用“下载但不重新初始化”。
2. 在现有块中插入新定义的变量。
3. 加载块到控制器中。

优势

- 重新加载新定义的变量，不中断运行过程。控制器保持“RUN”模式。

属性

- 下载但不重新初始化只适用于优化的块。
- 初始化新定义的变量。现有的变量保留其当前值。
- 一个有预留的块在控制器中需要更多的内存空间。
- 内存储备取决于控制器的工作内存。然而，最大为 2 MB。
- 假设已经为了块定义了内存预留。
- 默认下，内存预留为 100 byte。
- 内存预留是为每个块单独定义的。
- 块可以被可变地扩展。

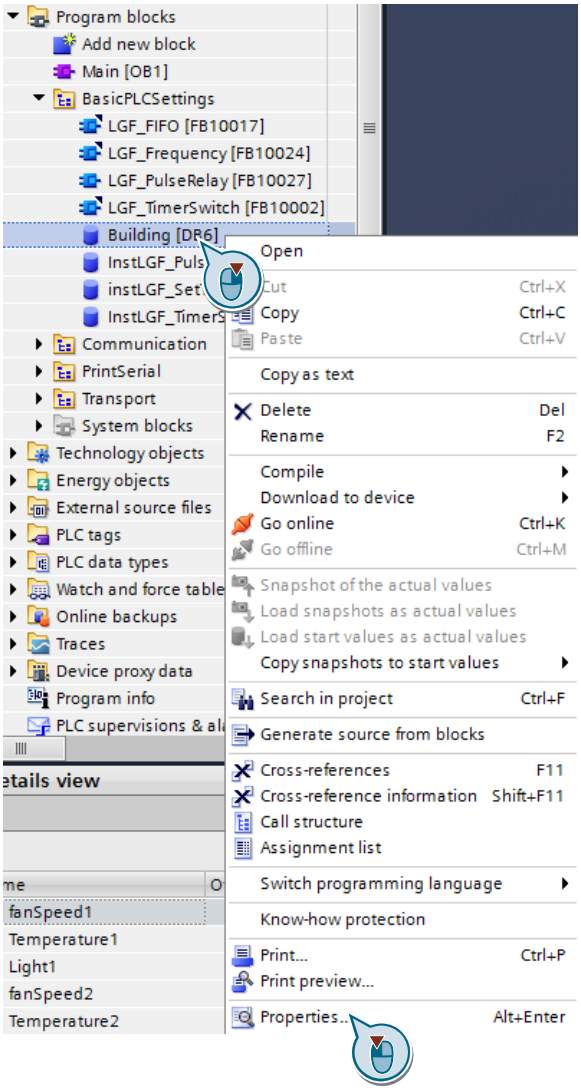
建议

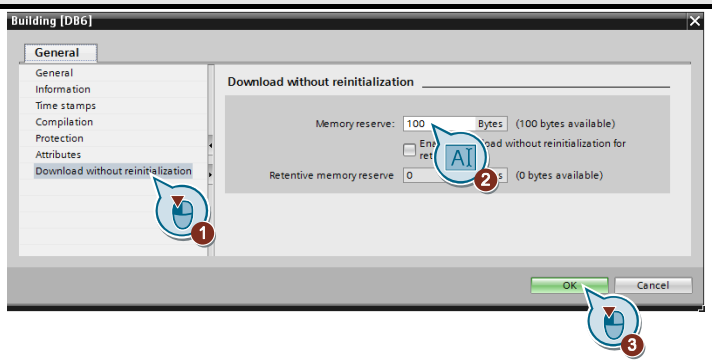
- 为调试期间要扩展的块(例如测试块)定义内存储备。由于现有变量的实际值仍然存在，因此调试过程不会受到下载的干扰。

示例：设置块上的内存预留

下表描述了如何为下载但不重新初始化设置内存预留。

表 3 -17：设置内存预留

步骤	说明
1.	在项目树中右键点击任意优化块，并选择属性。 

步骤	说明
2.	 点击下载但不重新初始化。 3. 为“内存预留”输入想要的内存预留值。 4. 点击 OK 确认。

注意

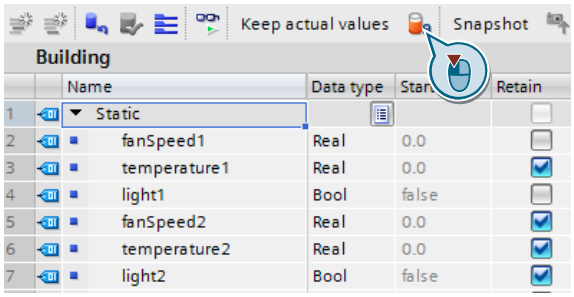
您还可以为 TIA 博途中新块的内存预留大小设置一个默认值。

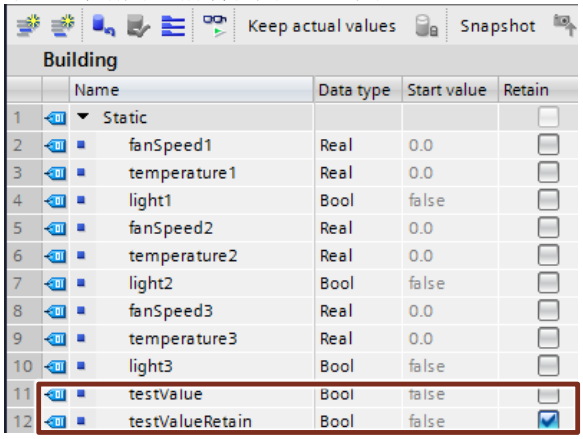
在菜单栏中，选择“选项-设置”，再选择“PLC 编程-常规-下载但不重新初始化”。

示例：下载但不重新初始化

下面的示例显示如何在不重新初始化的情况下进行下载。

表 3 -18：未初始化地加载

步骤	说明
1.	前提条件：必须设置内存储备(参见上面)。
2.	打开，例如优化的全局数据块。
3.	点击“激活存储区预留”按钮，用“确定”确认对话框。 

步骤	说明
4.	添加一个新变量(保持性变量也可以)。 
5.	将块下载到控制器。
6.	结果： • 块的实际值保持不变。

注意

更多信息可以在 TIA 博途网站的在线帮助“加载块扩展而无需重新初始化”下找到。

了解更多信息，请参阅以下条目：

STEP 7-1500(TIA 博途)中全局数据块的声明表是如何结构化的?
https://support.industry.siemens.com/cs/ww/en/view/68015630

3.2.9 块的可重用性

块概念提供了许多以结构化和有效的方式进行编程的选项。

优势

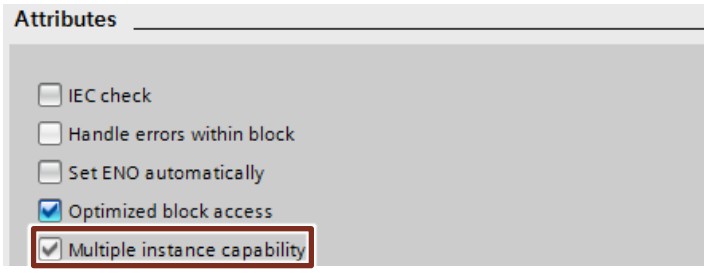
- 块可以普遍用于用户程序的任何位置。
- 块可以在不同的项目中普遍使用.
- 当每个块接收到一个独立的任务时，一个清晰的、结构良好的用户程序就会自动创建。
- 显著减少的错误来源。
- 可以进行简单的错误诊断。

建议

如果您想重复使用该块，请注意以下建议：

- 总是将块视为封装的功能。也就是说，每个块代表整个用户程序中完成的部分任务。
- 使用多个循环主 OB 块对工厂部件进行分组。
- 总是通过它的接口而不是实例来执行数据交换（见章节 3.4.1 块接口的数据交换）。
- 不使用项目特定数据，避免以下块内容：
  - 访问全局数据块和使用单一背景数据块
  - 访问变量
  - 访问全局常量
- 可重复使用的块与库的专有知识保护块有相同的需求。这就是为什么您必须基于“多重实例能力”块属性检查块的可重用性。在检查之前编译代码块。

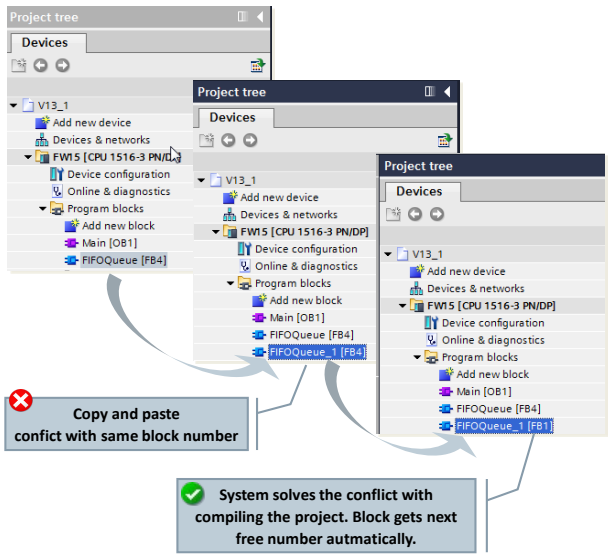
图 3 -19：块的属性



3.2.10 块的自动编号

对于内部程序，所需块的编号由系统自动分配(在块属性中设置)。

图 3 -20：块的自动编号



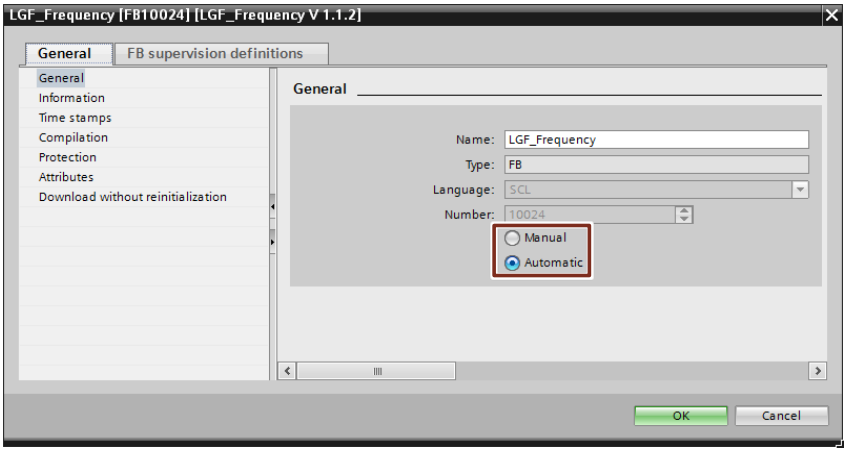
优势

- 冲突的块号，例如，由于复制导致的冲突会在 TIA 博途的编译期间自动删除。

建议

- 保持现有设置“自动的”不变。

图 3 -21：在块中设置



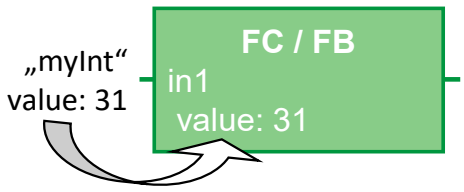
3.3 块的接口类型

FB 和 FC 有三种不同的接口类型：In、InOut 和 Out。通过这些接口类型，参数被提供给块。参数被处理并在块中再次输出。InOut 参数用于将数据传输到被调用块以及返回结果。数据的参数传输有两种不同的选项。

3.3.1 按值调用

调用块时，实际参数的值被复制到块的形参上。为此，在被调用块中提供额外的内存。

图 3 -22：值的传递



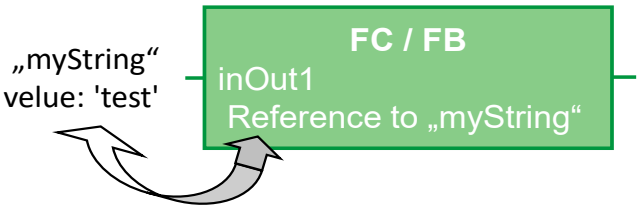
属性

- 每个块显示与传输的参数相同的行为。
- 调用块时复制值。

3.3.2 按引用调用

当调用该块时，一个引用被传递到实际参数的地址。为此，不需要额外的内存。

图 3 -23：引用实际参数(指向参数数据存储的指针)



属性

- 每个块显示与引用的参数相同的行为。
- 实际参数在块被调用时被引用，即访问时，直接读取或写入实际参数的值。

建议

- 通常使用 InOut 接口类型的结构化变量(如 ARRAY，STRUCT，STRING，type…)，以避免不必要地扩大所需的数据内存。

3.3.3 参数传递概述

下表概述了具有基本数据类型或结构化数据类型的 S7-1200/1500 块参数是如何传输的。

表 3 -24：参数传递概述

块类型 / 形式参数		基本数据类型	结构数据类型
FC	Input	复制	引用

块类型 / 形式参数		基本数据类型	结构数据类型
	Output	复制	引用
	InOut	复制	引用
FB	Input	复制	复制
	Output	复制	复制
	InOut	复制	引用

注意

当调用块时传输具有“非优化访问”属性的优化数据时，它通常作为副本传输。当块包含许多结构化参数时，这可能会迅速导致块的临时存储区域(本地数据栈)溢出。

这可以通过为两个块设置相同的访问类型来避免 (2.6.5 优化访问和非优化访问的块之间的参数传递)。

3.4 存储概念

对于 STEP 7，全局存储区域和本地存储区域之间通常存在差异。全局存储区域对于用户程序中的每个块都是可用的。本地存储区域仅在各自的块内可用。

3.4.1 块接口的数据交换

如果您封装函数并仅通过接口对块之间的数据交换进行编程，那么您显然具有优势。

优势

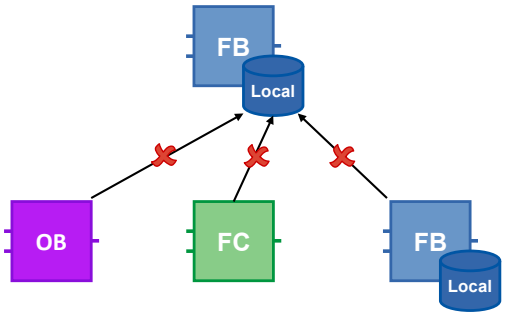
- 程序可以模块化地由带有部分任务的现成模块组成。
- 程序易于扩展和维护。
- 由于没有隐藏的交叉访问，程序代码更容易阅读和测试。

建议

- 如果可能的话，只使用本地变量。因此，您可以普遍地以模块化的方式使用这些块。

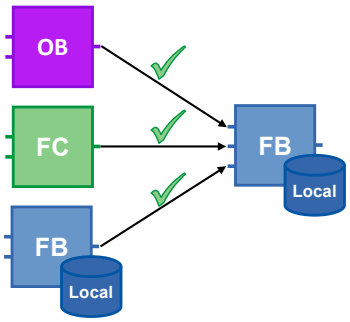
- 通过块接口(In，Out，InOut)使用数据交换，从而保证块的可重用性。
- 只使用背景数据块作为各自功能块的本地存储。其他块不应该写入背景数据块。

图 3 -25：避免访问背景数据块



如果只使用块接口进行数据交换，可以保证所有块都可以独立使用。

图 3 -26：块接口作为数据交换



3.4.2 全局存储

当存储可以从用户程序的任何位置访问时，就称为全局存储。有硬件相关的存储(例如：位存储、时间、计数器等)和全局数据块。对于依赖于硬件的存储区域，存在这样的危险：程序可能无法移植到任何控制器，因为那里的区域可能已经被使用了。这就是为什么应该使用全局数据块而不是依赖于硬件的存储区域。

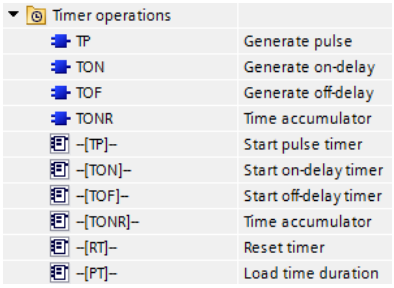
优势.

- 用户程序可以通用使用，独立于硬件。
- 用户程序模块化配置，无需为不同用户划分位存储区域。
- 优化的全局数据块显然比因为兼容性原因而没有优化的位存储区更强大。

建议

- 不要使用任何位存储区，而是使用全局数据块。
- 避免硬件依赖的内存，例如，时钟存储器或计数器。在多重实例中使用 IEC 计数器和计时器(见章节 3.2.5 多重实例)。IEC 计时器可以在“指令-基本指令-定时器操作”中找到。

图 3 -27：IEC 计时器



3.4.3 本地存储

- 静态变量
- 临时变量

建议

- 如果值是需要在下一个周期使用使用静态变量。
- 使用临时变量作为当前周期的中间存储。临时变量的访问时间比静态变量短。
- 如果一个 Input/Output 变量被频繁访问，使用一个临时变量作为中间存储来节省运行时间。

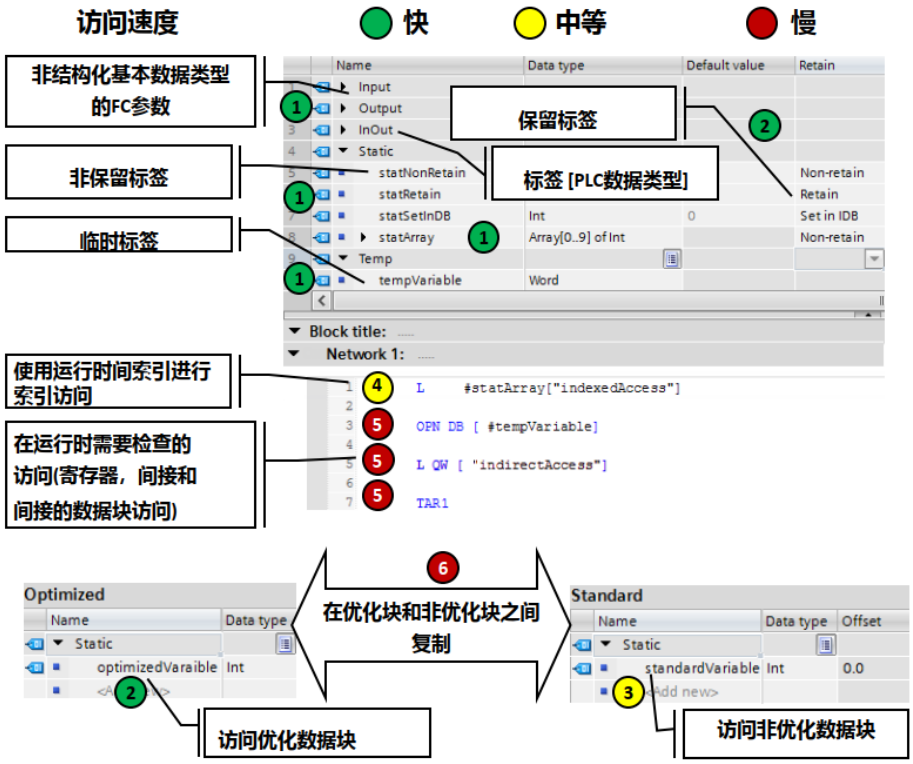
注意

优化块：在每次块调用中使用默认值(S7-1500/S7-1200 固件 V4 或更高版本)初始化临时变量。
非优化的块：临时变量在块的每次调用时都没有定义。

3.4.4 存储区域访问速度

STEP 7 提供不同的存储访问选项。由于系统相关的原因，对不同存储区域的访问有快有慢。

图 3 -28：不同的存储访问



由高到低的 S7-1200/1500 访问速度

1. 优化块:临时变量，FC 和 FB 参数，非保持型静态变量，变量[PLC 数据类型]。
2. 对编译的访问权限已知的优化块：
   - 保持型 FB 变量。
   - 优化的全局数据块。
3. 对非优化块的访问。
4. 在运行时计算数组索引的间接寻址(例如 Motor [i])。
5. 需要在运行时进行检查的访问。
   - 访问运行时创建或间接打开的数据块(例如 OPN DB[i])。
   - 寄存器访问或间接内存访问。
6. 在优化块和非优化块之间复制结构(字节数组除外)。

3.5 保持性

在电源故障的情况下，控制器用其缓冲能量将保持数据从控制器的工作存储器复制到非易失性存储器。重新启动控制器后，使用保持的数据恢复程序进程。根据不同的控制器，可保持的数据量大小不同。

表 3 -29：S7-1200/1500 的保持性存储器

控制器	可用于位存储器，定时器，计数器，数据块和工艺对象的保持性存储区
CPU 1211C,1212C, 1214C, 1215C, 1217C	14 kByte
CPU 1511-1 PN	88 kByte
CPU 1513-1 PN	88 kByte
CPU 1515-2 PN, CPU 1516-3 PN/DP	472 kByte
CPU 1518-4 PN/DP	768 kByte

表 3 -30：S7-1200 与 S7-1500 的不同

S7-1200	S7-1500
只能为位存储器设置保持性	可为位存储器、S7 定时器和 S7 计数器设置保持性

优势

- 保持性数据在控制器从 STOP 到 RUN 或在电源故障和控制器重启的情况下保持其值。

属性

对于优化的数据块的基本变量，可单独设置保持性。非优化数据块只能定义为完全保持或非保持。

保持性数据可以通过“内存复位”或“重置为出厂设置”的操作清除：

- 控制器的工作开关(MRES)
- 控制器显示屏
- 通过 STEP 7(TIA 博途）在线操作

建议

不要使用“在 IDB 中设置”。保持性数据总是设置在功能块中，而不是在背景数据块中。

“IDB 中的设置”增加程序序列的处理时间。总是在 FB 的接口中选择“非保持”或“保持”。

.

图 3 -31：程序编辑器（函数块接口）

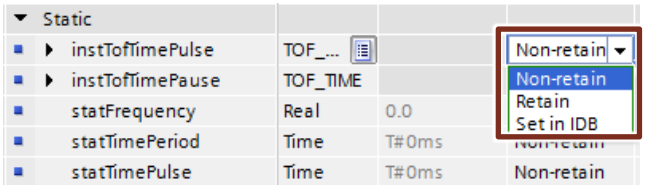
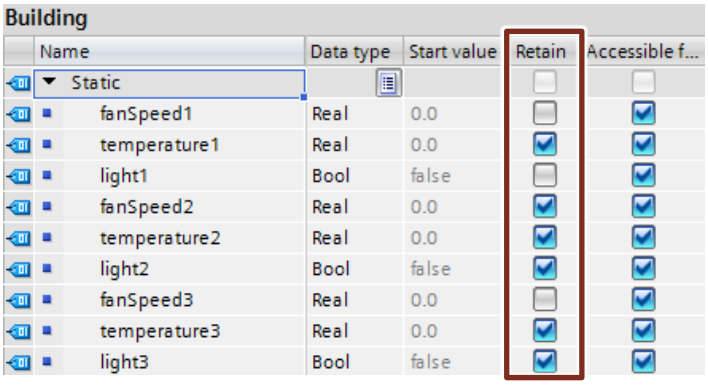


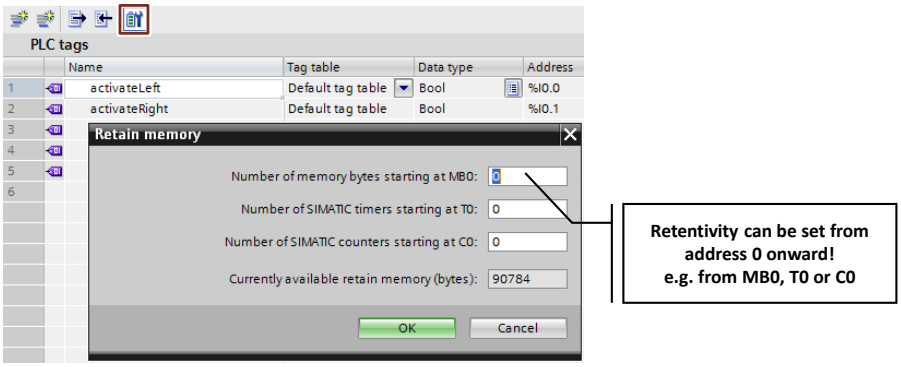
图 3 -32：程序编辑器（数据块）



示例：保持性 PLC 变量

保持数据的设置在 PLC 变量表、功能块、数据块中进行。

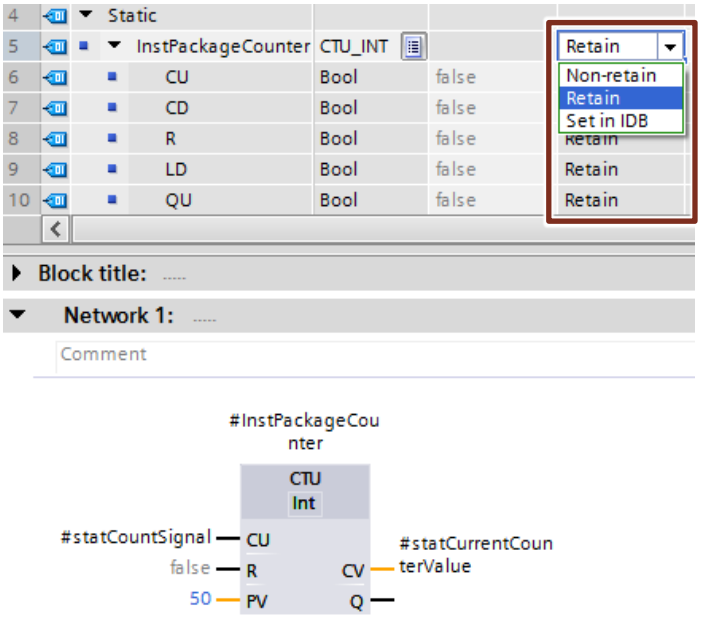
图 3 -33：PLC 变量表中保持性变量的设置



示例：保持性计数器

你也可以声明函数(计时器，计时器等)的实例的保持性。正如章节 3.2.5 多重实例所描述的，应该总是把这样的函数编程为多重实例。

图 3 -34：作为多重实例的保持性计数器



注意

如果 PLC 上的保持内存不足，则可以以数据块的形式存储数据，这些数据块只位于 PLC 的装载内存中。下面以 S7-1200 为例进行说明。这种编程也适用于 S7-1500。

更多资料可参阅下列各项：

如何配置 STEP 7(TIA 博途)中的数据块与 S7-1200 的“仅存储在装载内存中”属性？
https://support.industry.siemens.com/cs/ww/en/view/53034113

针对 SIMATIC S7-1200 和 S7-1500 的持久数据使用配方功能。
https://support.industry.siemens.com/cs/ww/en/view/109479727

3.6 符号寻址

3.6.1 符号寻址而非绝对寻址

TIA 博途为符号编程进行了优化。这带来了许多好处。由于使用符号寻址，您可以在编程时不需要关注内部数据存储。控制器负责处理数据的最佳存储位置。因此，您可以完全专注于应用程序任务的解决方案。

优势

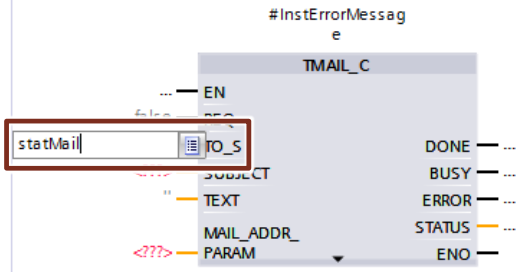
- 通过符号变量名称更容易读取程序。
- 自动更新用户程序中所有使用位置的变量名称。
- 程序数据的内存存储不需要手动管理(绝对寻址)。
- 强大的数据访问功能。
- 不需要手动优化性能或程序大小。
- 自动完成快速符号输入。
- 由于类型安全(所有访问都检查数据类型的有效性)，减少了程序错误。

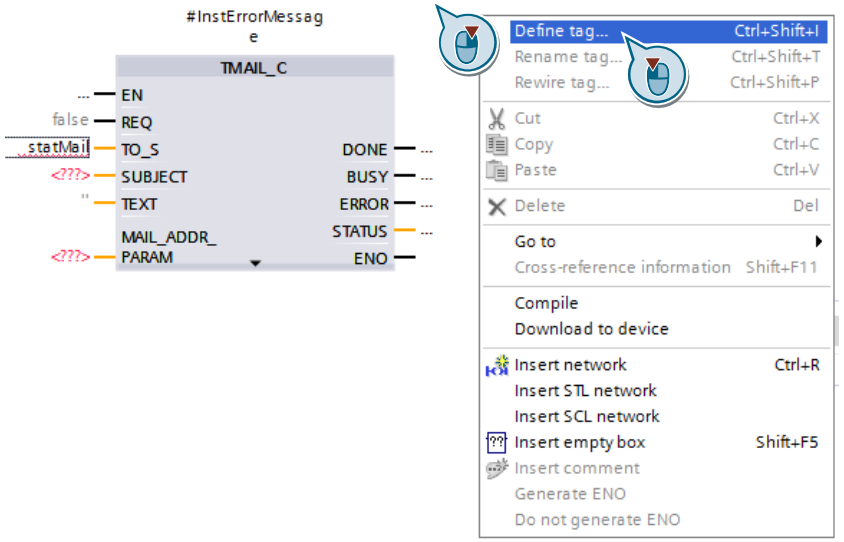
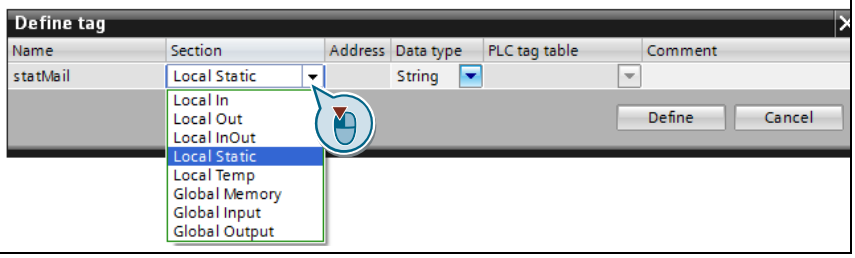
建议

- “不用担心数据的存储”
- “思考”的象征意义。为每个功能、变量或数据输入“描述性”名称，例如，Pump_boiler_1、heater_room_4 等。因此，创建的程序可以简单地读懂，而不需要很多注释。
- 给所有变量使用了直接的符号名称，然后右键来定义它们。

示例

表 3 -35：创建符号变量的示例

步骤	说明
1.	打开程序编辑器并打开任意块。
2.	在指令的输入处直接输入符号名称。 

步骤	说明
3.	在块旁边右击并在上下文菜单中选择“定义变量…” 
4.	定义变量 

如果您想在网络中定义多个变量，有一种简练的方法可以节省时间。首先，分配所有变量名称。然后用步骤 4 的对话框同时定义所有变量。

注意

更多资料可参阅下列条目：

在 STEP 7 (TIA 博途) 中使用 S7-1500 的符号寻址的优点是什么?
https://support.industry.siemens.com/cs/ww/en/view/67598995

3.6.2 ARRAY 数据类型和间接寻址访问

ARRAY 数据类型表示由数据类型的多个元素组成的数据结构。ARRAY 数据类型适用于存储配方、队列中的物料跟踪、循环过程获取、协议等。

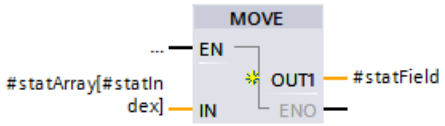
图 3 -36：包含 10 个 INT 数据类型元素的 ARRAY

Name	Data type
statArray	Array[0..9] of Int
statArray[0]	Int
statArray[1]	Int
statArray[2]	Int
statArray[3]	Int
statArray[4]	Int
statArray[5]	Int
statArray[6]	Int
statArray[7]	Int
statArray[8]	Int
statArray[9]	Int

可以通过索引间接访问 ARRAY 中的各个元素 (`array ["index"]`).

图 3 -37：间接字段访问

KOP / FUP:



SCL:

```
1  #statField := #statArray[#statIndex];
```

优势

- 通过 ARRAY 索引轻松访问。
- 不需要复杂的指针创建。
- 能够快速创建和扩展。
- 适用于所有编程语言。

属性

- 结构化数据类型。
- 由固定数量的相同数据类型的元素组成的数据结构。
- 也可以创建多维 ARRAY。
- 可以间接访问运行时变量，并在运行时进行动态索引计算。

建议

- 使用 ARRAY 进行索引访问，而不是指针（例如 ANY 指针）。这使得程序更容易读懂，因为使用符号 ARRAY 更有意义。
- 为运行变量使用 DINT 数据类型作为临时变量，以获得最高的性能。
- 用“MOVE_BLK”指令将 ARRAY 的部分复制到另一个 ARRAY 中。
- 使用“GET_ERR_ID”指令来捕获 ARRAY 中的访问错误。

注意

更多资料可参阅下列各项：

在 S7-1500 中如何用变量索引实现一个数组访问?
https://support.industry.siemens.com/cs/ww/en/view/67598676

在 STEP 7(TIA 博途)中如何安全地和间接地引址？
https://support.industry.siemens.com/cs/ww/en/view/97552147

在 STEP 7 (TIA 博途)中,如何在 S7-1500/S7-1200 中进行"Array of Bool" 和 "Word"之间的数据类型转换?
https://support.industry.siemens.com/cs/ww/en/view/108999241

3.6.3 形参 Array [*] (V14 或更高版本)

通过形参 Array[*]，可以将可变长度的 Array 传递给函数和函数块。
使用 “LOWER_BOUND” 和 “UPPER_BOUND” 指令来确定数组的上下限。

优势

- 块可以处理不同长度灵活的 Array。
- 基于全符号编程的最佳可读性。
- 不再需要为不同长度的 Array 编写指针程序。

示例

图 3 -38：初始化不同的数组

3.6.4 STRUCT 数据类型和 PLC 数据类型

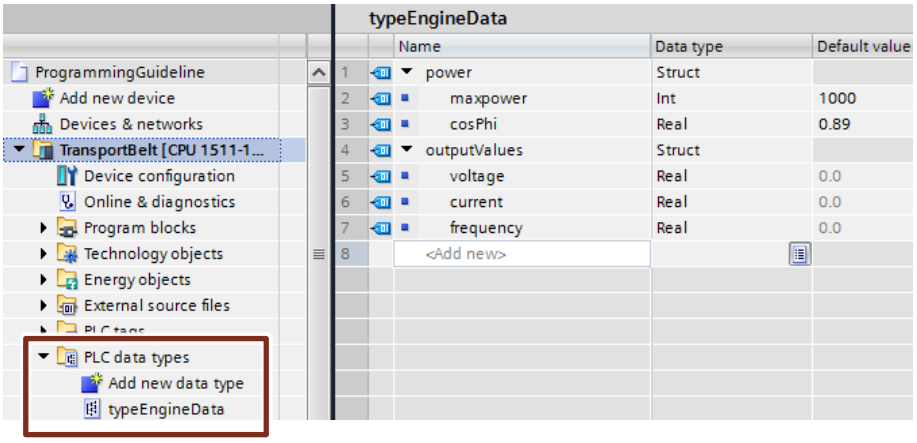
STRUCT 数据类型表示由不同数据类型的元素组成的数据结构。结构的声明在各自的块中执行。

图 3 -39：使用具有不同数据类型的元素的结构

Name	Data type	Default value
statEngineData	Struct	
power	Struct	
maxpower	Int	1000
cosPhi	Real	0.89
<Add new>		
outputValues	Struct	
voltage	Real	0.0
current	Real	0.0
frequency	Real	0.0
<Add new>		

与结构相比，PLC 数据类型在 TIA 博途中基于控制器定义，可以集中更改。所有使用位置都会自动更新。
使用 PLC 数据类型前，在项目导航的“PLC 数据类型”文件夹中声明。

图 3 -40：PLC 数据类型



优势

- PLC 数据类型的变化会自动更新到用户程序中的所有使用位置。
- 通过几个块之间的块接口进行简单的数据交换。
- 在 PLC 数据类型中，可以声明具有定义长度的 STRING 变量(例如 String[20])。在 TIA V14 中，也可以使用一个全局常量来表示长度(例如 String[LENGTH])。
  如果 STRING 变量没有定义长度，则该变量的最大长度为 254 个字符。

属性

- PLC 数据类型总是以 WORD 限制结束(见下图)。
- 请考虑这个系统属性当…
  - 在 I/O 区域使用结构(见章节 3.6.5 访问具有 PLC 数据类型的 I/O 区域)。
  - 使用带有 PLC 数据类型的框架进行通信。
  - 为 I/O 使用带有 PLC 数据类型的参数记录。
  - 使用非优化块和绝对寻址。

图 3 -41：PLC 数据类型总是以 WORD 限制结束

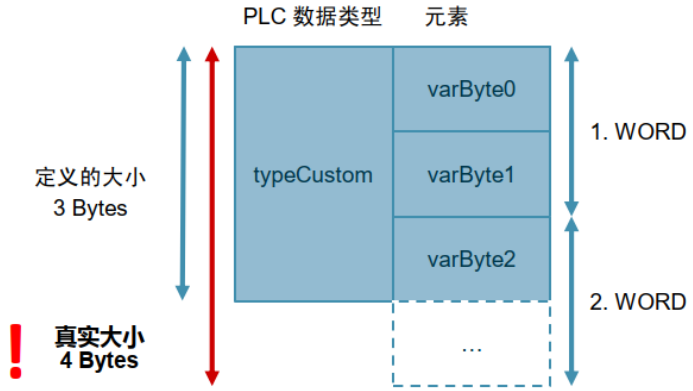
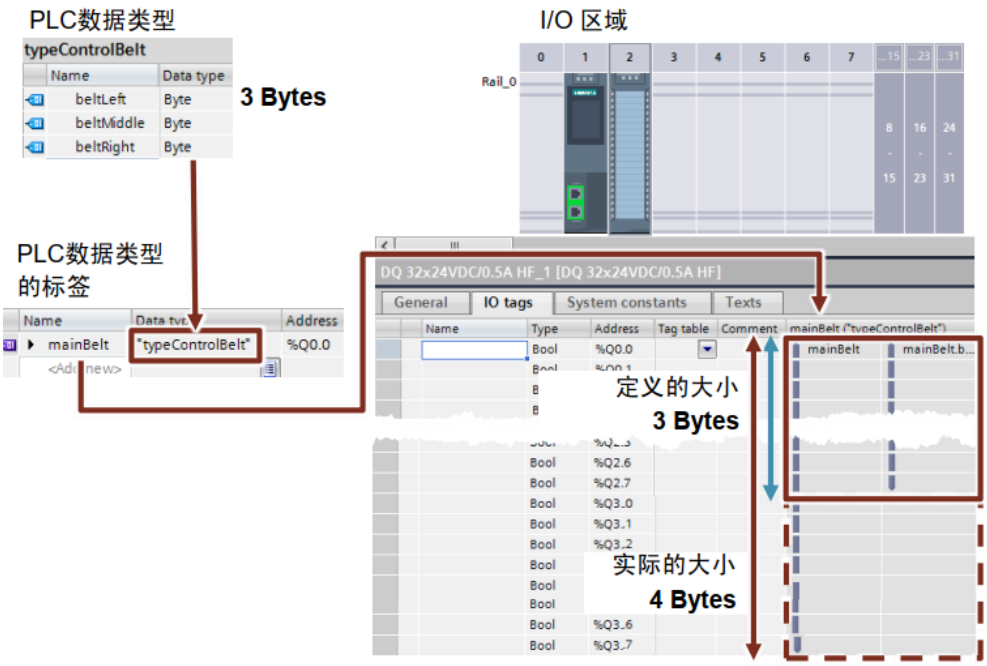


图 3 -42：I/O 区域的 PLC 数据类型



建议

- 使用 PLC 数据类型来汇总几个相关的数据，例如，框架或电机数据(设定值，速度，旋转方向，温度等)。

- 在用户程序的多种用途中，总是使用 PLC 的数据类型而不是结构。
- 使用 PLC 的数据类型构造数据块。
- 使用 PLC 的数据类型来指定数据块的结构。PLC 数据类型可用于任意数量的数据块。您可以轻松方便地创建任意数量的相同结构的数据块，并在 PLC 数据类型上集中调整它们。

注意

更多信息可以在以下条目中找到：

适用于 STEP 7 (TIA Portal) 和 SIMATIC S7-1200/S7-1500 的 PLC 数据类型 (LPD) 库
https://support.industry.siemens.com/cs/ww/en/view/109482396

如何将结构初始化为 S7-1500 STEP 7（TIA 博途）的优化内存区域?
https://support.industry.siemens.com/cs/ww/en/view/78678760

如何为 S7-1500 控制器创建 PLC 数据类型？
https://support.industry.siemens.com/cs/ww/en/view/67599090

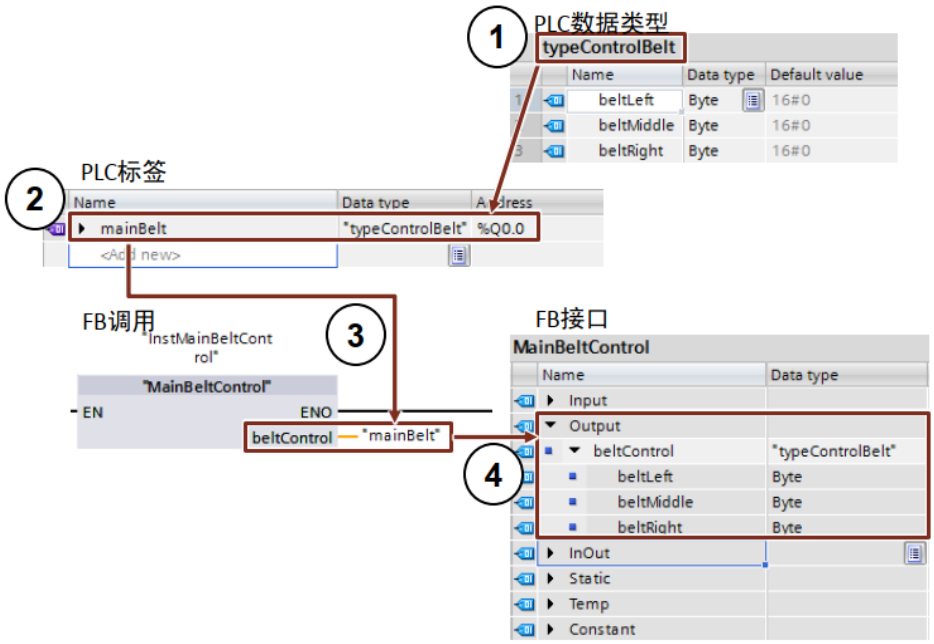
在 STEP 7（TIA 博途）中，您如何应用自己的数据类型（UDT）?
https://support.industry.siemens.com/cs/ww/en/view/67582844

对于 S7-1500，为什么当调用一个功能块时应该使用整个结构代替大量的单个元素来传递参数？
https://support.industry.siemens.com/cs/ww/en/view/67585079

3.6.5 访问具有 PLC 数据类型的 I/O 区域

使用 S7-1500 控制器，您可以创建 PLC 数据类型，并使用它们进行结构化和符号访问输入和输出。

图 3 -43：访问具有 PLC 数据类型的 I/O 区域



1. PLC 数据类型，包含所有需要的数据。
2. 创建的 PLC 数据类型的 PLC 变量类型和 I/O 数据区域的起始地址(%Ix.0 或%Qx.0，例如%I0.0，%Q12.0…)。
3. 将 PLC 变量作为实际参数传递到功能块。
4. 功能块的输出类型为创建的 PLC 数据类型。

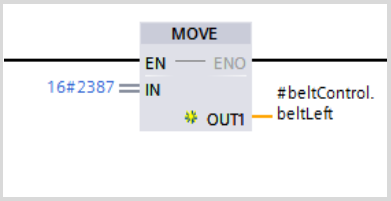
优势

- 编程效率高。
- PLC 数据类型易于多种可用性。

建议

- 使用 PLC 数据类型访问 I/O 区域，例如，符号性地接收和发送驱动器报文。

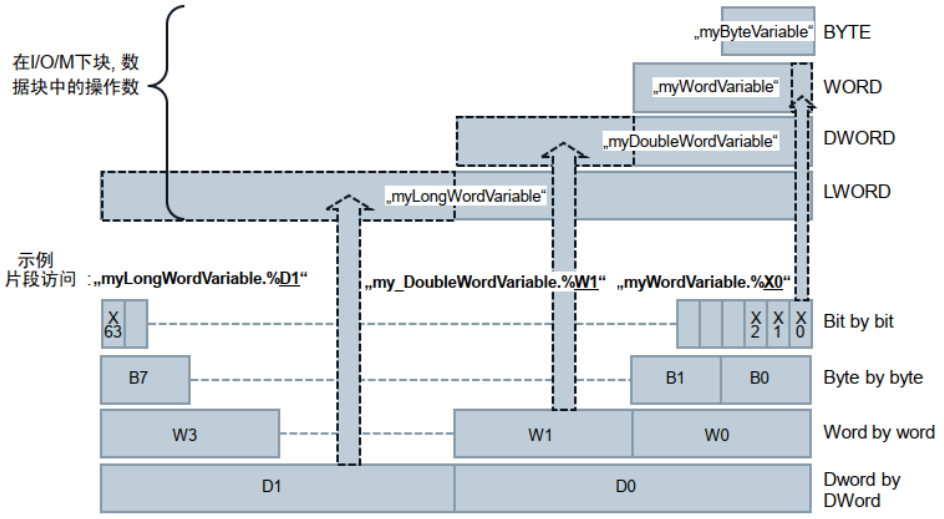
注意 在用户程序中也可以直接访问一个 PLC 数据类型变量中的各个元素：



3.6.6 片段访问

对于 S7-1200/1500 控制器，可以访问 Byte、Word、DWord 或 LWord 数据类型变量的存储区。将一个存储区域(如 byte 或 word)划分为一个较小的存储区域(如 Bool)也称为片段。下图显示了对操作数的 bit、byte 和 word 的访问。

图 3 -44：bit，byte，word，DWord 符号的片段访问



优势

- 编程效率高。
- 在变量声明中不需要额外的定义。
- 易于访问(例如控制位)。

建议

- 使用 AT 结构的片段访问，而不是访问操作数中的某些数据区域。

注意

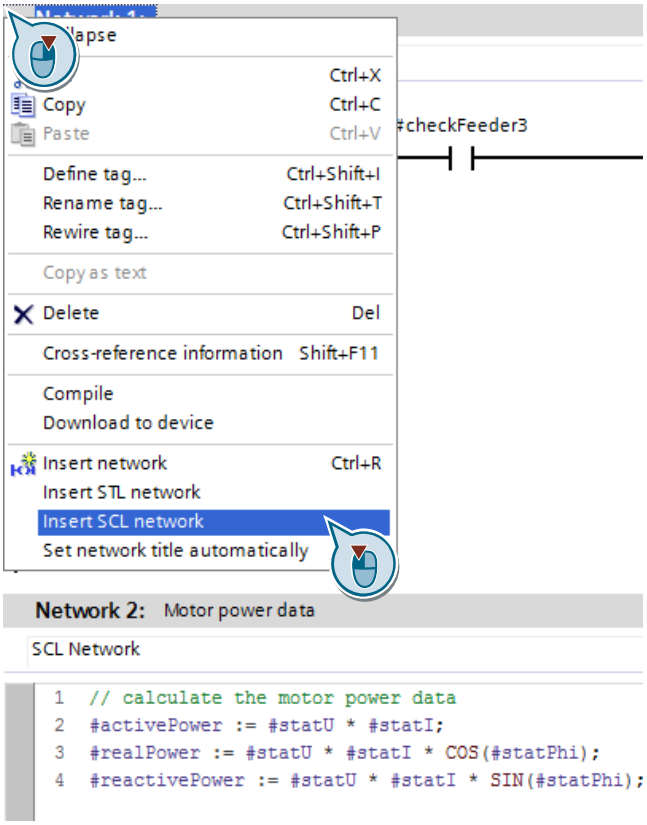
更多信息可以在以下条目中找到：

在 STEP 7（TIA 博途）中，您如何以符号的方式逐位、逐字节或逐字访问非结构化数据类型？
https://support.industry.siemens.com/cs/ww/en/view/57374718

3.6.7 LAD 和 FBD 的 SCL 网络(V14 及更高版本)

您可以在 LAD 和 FBD 中使用 SCL 网络进行计算编程，而其他编程在 LAD 和 FBD 指令中进行。

图 3 -45：插入 SCL 网络



优势

- 通过高效编程节省时间。

- 得益于符号编程，代码清晰。

属性

- 支持所有 SCL 指令。
- 支持注释。

建议

- 在 LAD 和 FBD 中使用 SCL 网络进行数学计算，而不是使用 ADD、SUB 等指令。

3.7 库

使用 TIA Portal，您可以从不同的项目元素建立独立的库，这些库可以轻松重复使用。

优点

- 在 TIA Portal 中组态配置的数据的简单存储：
  - 完整的设备（控制器、HMI、驱动器等）
  - 块、变量表、PLC 数据类型、监视表等。
  - HMI 画面、HMI 变量、脚本等
- 通过库进行跨项目交互
- 库元素的集中更新功能
- 库元素的版本控制
- 通过系统支持的依赖关系考虑使用控制块时更少的错误源

建议

- 创建模板副本以轻松重复使用块、硬件配置、HMI 画面等。
- 为系统支持的可重用性库元素创建类型：
  - 块的版本控制
  - 所有使用位置的集中更新功能
- 使用全局库与其他用户进行交互，或作为多个用户同时使用的中央存储。
- 配置全局库的存储位置，以便在启动 TIA Portal 时自动打开。
  如需更多信息，请访问：
  https://support.industry.siemens.com/cs/ww/en/view/100451450

注意

更多信息可以在以下条目中找到：

STEP 7（TIA 博途）和 WinCC（TIA 博途）的哪些元素可以作为类型或主副本存储在库中？
https://support.industry.siemens.com/cs/ww/en/view/109476862

如何在 STEP 7（TIA 博途）中打开具有读写访问权限的全局库？
https://support.industry.siemens.com/cs/ww/en/view/37364723

3.7.1 库类型和库元素

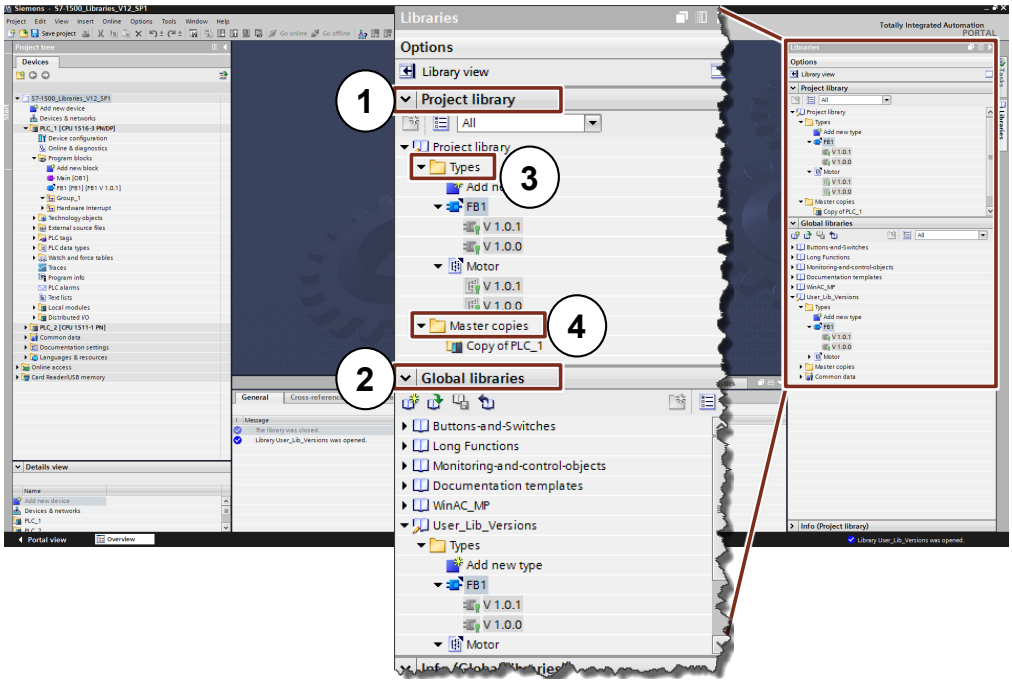
通常有两种不同类型的库：

- “项目库”
- “全局库”

内容由两种存储类型组成：

- “类型”
- “模板副本”

图 3 -46：TIA Portal 中的库



(1) “项目库”

- 集成在项目中，与项目一起管理
- 允许项目内可重复使用

(2) “全局库”

- 独立库
- 可在多个项目中使用

库包括两种不同类型的库元素存储：

(3) “模板副本”

- 库中配置元素的副本（例如块、硬件、PLC 变量表等）
- 副本不与项目中的元素相关联。
- 模板副本也可以由几个配置元素组成。

(4) “类型”

- 类型与您在项目中的使用位置相关联。当类型发生变化时，项目中的使用位置可以自动更新。
- 支持的类型是控制块（FC、FB）、PLC 数据类型、HMI 画面、HMI faceplate、HMI UDT、脚本）。
- 下级元素是自动类型化的。
- 类型是版本化的：可以通过创建更新的版本来进行更改。
- 控制器中只能有一个已使用类型的版本。

3.7.2 类型概念

类型概念允许创建可在多个工厂或机器中使用的标准化自动化功能。类型概念提供了版本控制和更新功能。

可以在用户程序中使用库中的类型。这提供了以下优点

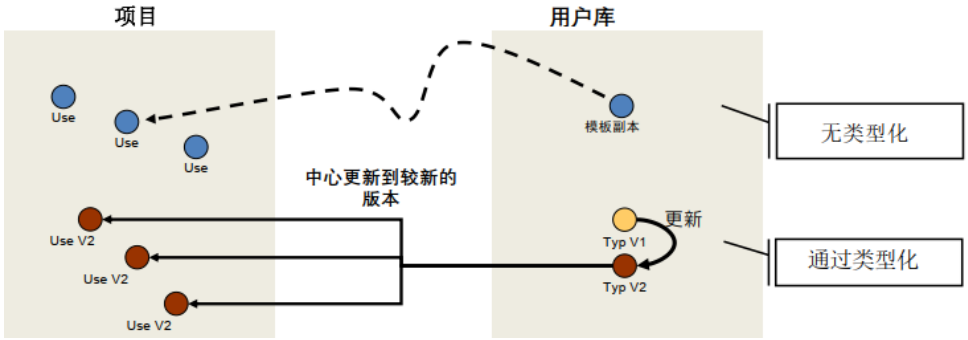
优点

- 项目中所有使用位置的集中更新
- 不可对类型的使用位置进行不必要的修改。
- 系统通过阻止不需要的删除操作来保证类型始终保持一致。
- 如果一个类型被删除，用户程序中的所有使用位置都会被删除。

特性

通过使用类型，可以集中进行更改并在整个项目中更新它们。

图 3 -47：使用用户库进行类型化



- 始终标记类型以便更好地识别

3.7.3 CPU 和 HMI 中典型对象之间的差异

在控制器和 HMI 中典型对象之间存在系统相关的差异：

表 3 -48：控制器和 HMI 的类型差异

控制器	HMI
从属控制元素被类型化。	从属 HMI 元素**不会被类型化**。
从属控制元素会被实例化。	从属 HMI 元素**不会**被实例化。
在**测试环境**中编辑控制元素。	HMI 图像和 HMI 脚本在测试环境中进行编辑。Faceplate 和 HMI UDT 直接在库中编辑，**无需测试环境**。

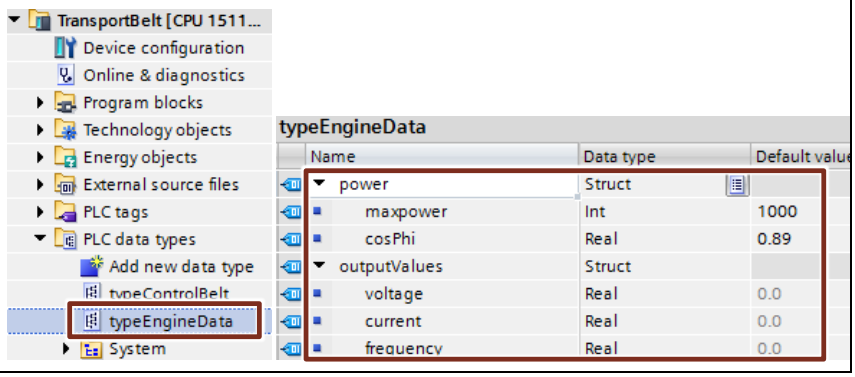
可以在以下示例中找到有关处理库的更多信息。

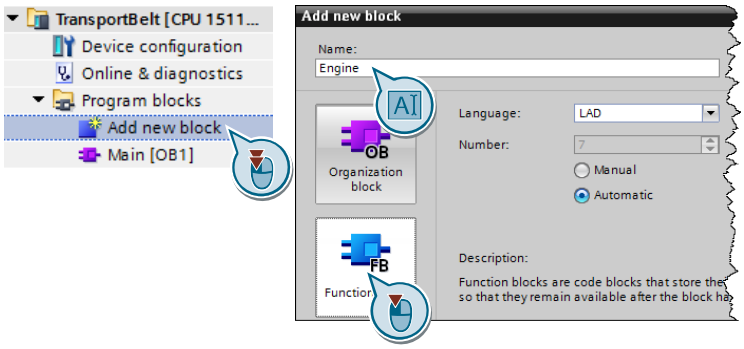
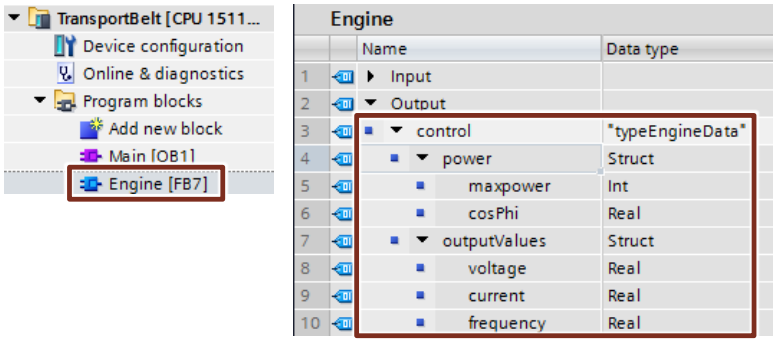
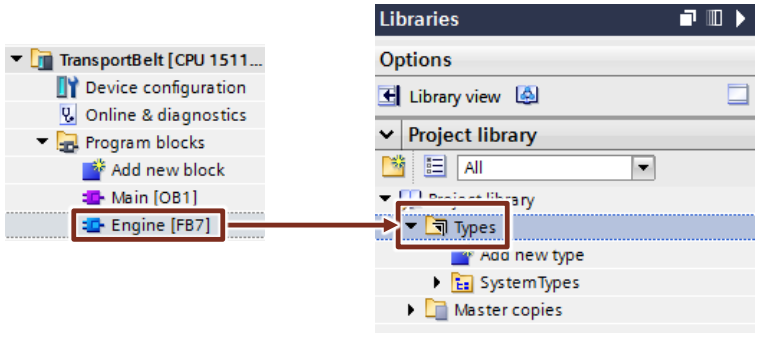
3.7.4 块的版本控制

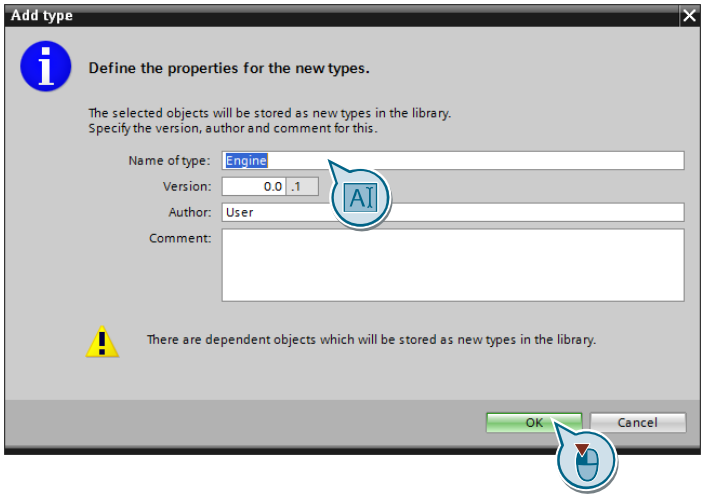
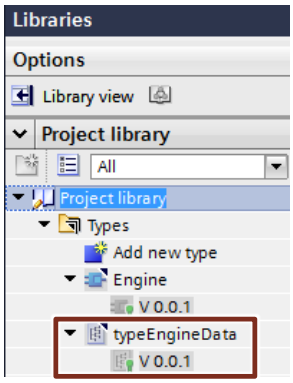
示例：创建类型

以下示例向您. 展示了库的基本功能如何与类型一起使用。

表 3 -49：创建类型

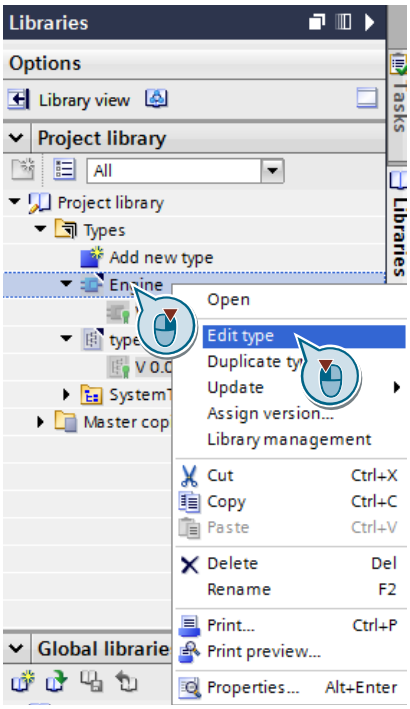
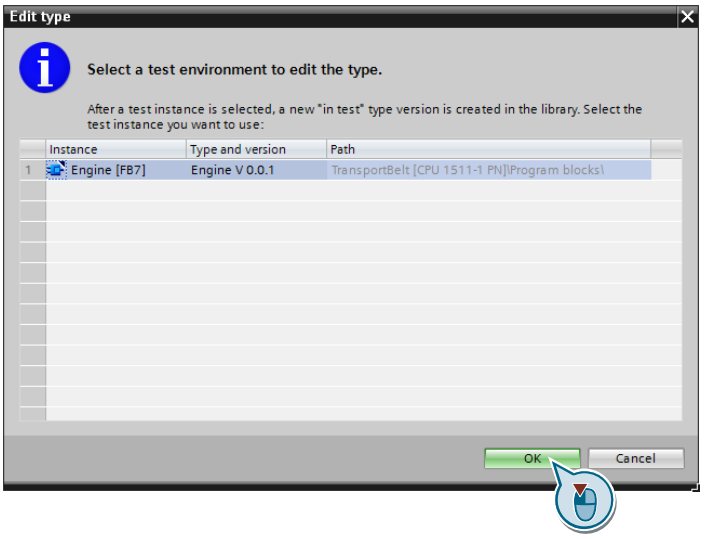
步	操作说明
1.	使用“添加新数据类型”创建一个新的 PLC 数据类型并创建一些变量。之后作为从属类型。 

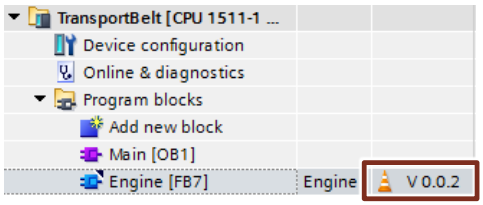
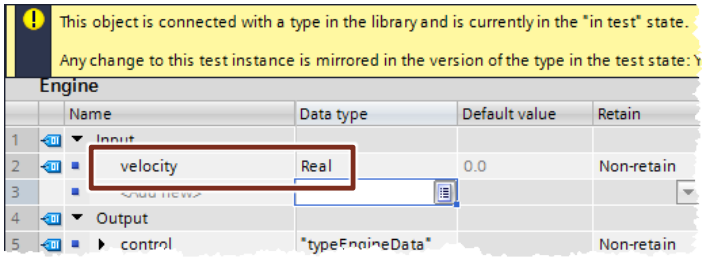
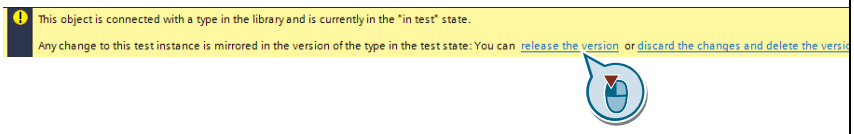
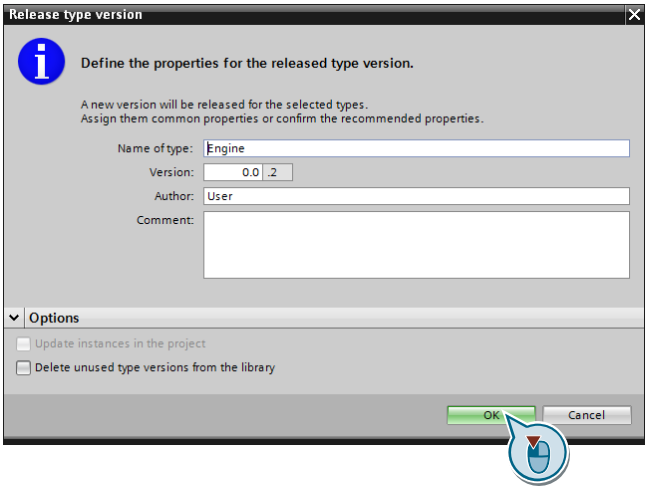
步	操作说明
2.	使用“添加新块”创建一个新功能块。这是更高级别的类型。 
3.	定义输出变量为创建的数据类型。PLC 数据类型从属于功能块。 
4.	通过拖放将功能块拖放到项目库中的“类型”文件夹中。 

步	操作说明
5.	可选择分配：输入名称、版本、作者和注释，然后单击“确定”确认对话框。 
6.	从属的 PLC 数据类型也自动存储在库中。 

示例：更改类型

表 3 -50：更改类型

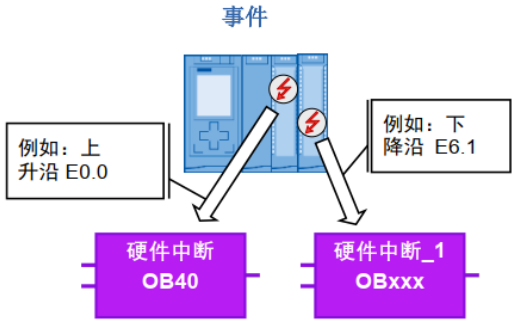
步	操作说明
1.	右键单击“项目库”中的块并选择“编辑类型” 
2.	选择将哪个控制器用作测试环境，然后单击“确定”确认对话框。  如果项目中有多个控制器使用选定的块，则必须选择一个控制器作为测试环境。

步	操作说明
3.	块打开。一个新版本的块被创建。 
4.	添加输入变量。 在这个地方，您可以选择通过将项目加载到控制器上来测试块上的更改。完成测试块后，继续执行以下步骤。 
5.	点击“发布版本”按钮。 
6.	一个对话框打开。在这里您可以存储与版本相关的注释。单击“确定”确认对话框。  如果项目的不同控制器中有块的多个使用位置，可以同时更新：“更新项目中的实例”。 如果不再需要旧版本的元素，您可以通过单击“从库中删除未使用的类型版本”来删除它们

3.8 提高硬件中断的性能

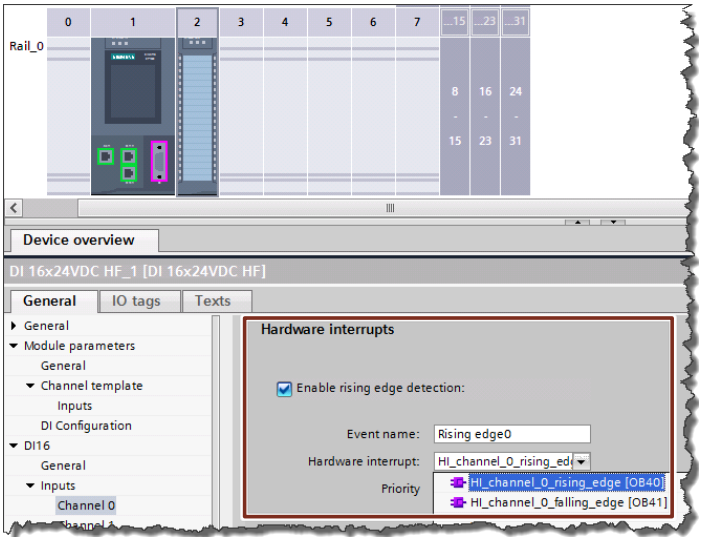
用户程序的处理可能会受到硬件中断等事件的影响。当您需要控制器对硬件事件（例如数字输入模块通道的上升沿）做出快速响应时，组态硬件中断。对于每个硬件中断，可以编写一个单独的 OB。发生硬件中断时，控制器的操作系统会调用此 OB。因此，控制器的循环被中断，在处理硬件中断后将继续。

图 3 -51：在硬件中断时调用 OB



在下图中，您可以看到数字量输入模块的硬件分配中的“硬件中断”配置。

图 3 -52：分配硬件中断



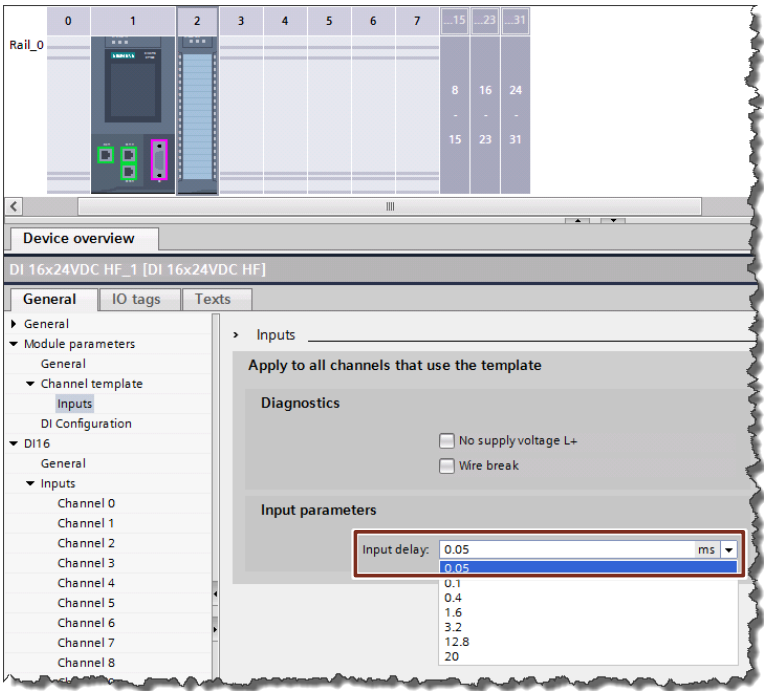
优点

- 对事件（上升沿、下降沿等）的快速系统响应
- 每个事件都可以启动一个单独的 OB。

建议

- 使用硬件中断以便对硬件事件的快速响应进行编程。
- 尽管编程了硬件中断，但如果系统响应速度还是不够快，可以进一步加快响应速度。在模块中设置尽可能小的“输入延迟”。只有在输入延迟已过时，才会对事件做出响应。输入延迟用于过滤输入信号，例如，补偿诸如抖动的故障。

图 3 -53：设置输入延迟



3.9 其他性能建议

在这里，您可以找到一些能够加快控制器程序处理速度的一般性建议。

建议

请注意以下有关对 S7-1200/1500 控制器进行编程以实现高性能的建议：

- LAD/FBD：禁用块的“评估 ENO”。这避免了运行时的测试。
- STL：不要使用寄存器，因为 S7-1500 出于兼容性原因才模拟地址和数据寄存器。

注意

更多信息可以在以下条目中找到：

如何禁用指令的 ENO 使能输出？
https://support.industry.siemens.com/cs/ww/en/view/67797146

如何提高 STEP 7（TIA 博途）和 S7-1200/S7-1500 CPU 的性能？
https://support.industry.siemens.com/cs/ww/en/view/37571372

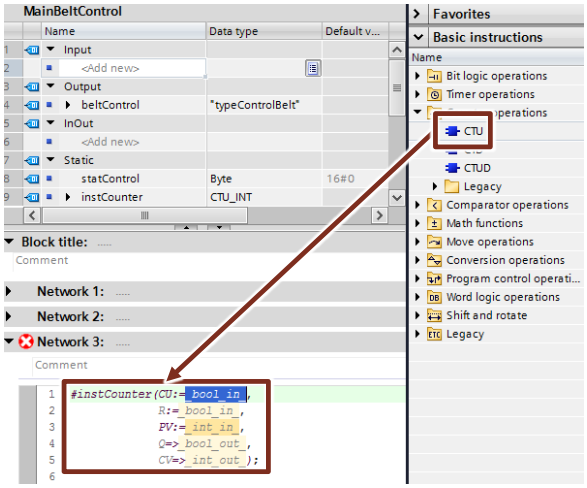
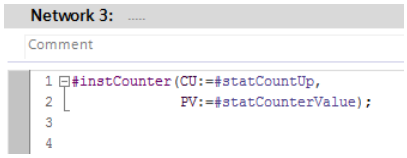
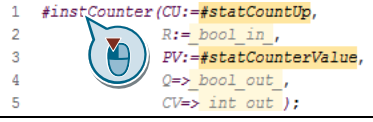
3.10 SCL 编程语言：提示和技巧

3.10.1 使用调用模板

编程语言的许多指令都提供了一个调用模板，其中包含现有形式参数的列表。

例子

表 3 -54：调用模板的简单扩展

步骤	操作说明
1.	将指令从库中拖到 SCL 程序中。编辑器显示完整的调用模板。 
2.	现在填写所需的参数“CU”和“PV”并使用“返回”按钮完成输入。
3.	编辑器会自动折叠调用模板。 
4.	如果想稍后再次编辑完整的调用，请按以下步骤操作。 在任意位置点击进入调用，然后点击“**CTRL+SHIFT+SPACE**”。现在处于“调用模板”模式。编辑器再次展开调用。可以使用“TAB”按键浏览参数。 
5.	注意：在“调用模板”模式下，文字为斜体。

3.10.2 哪些指令参数是强制性的？

如果正在展开调用模板，颜色编码将立即向您显示指令的哪些形式参数是可选的，哪些不是。强制参数标记为黑色。

3.10.3 使用整个变量名称进行拖放

在 SCL 编辑器中，您还可以使用拖放功能。对于变量名称，也支持拖放。如果您想用另一个变量替换另一个，请执行以下操作。

表 3 -55：在 SCL 中使用变量拖放

步	操作说明
1.	通过拖放将变量拖放到要替换的程序中的变量上。按住变量 1 秒以上再松开。  完整的变量被替换。

3.10.4 使用关键字 REGION（V14 或更高版本）进行结构化

SCL 代码可以用关键字 REGION 划分为不同区域。这些区域可以命名，也可以折叠和展开。

优点

- 更好的总览
- 即使在大型块中也能轻松定位
- 准备好的代码片段可以折叠。

特性

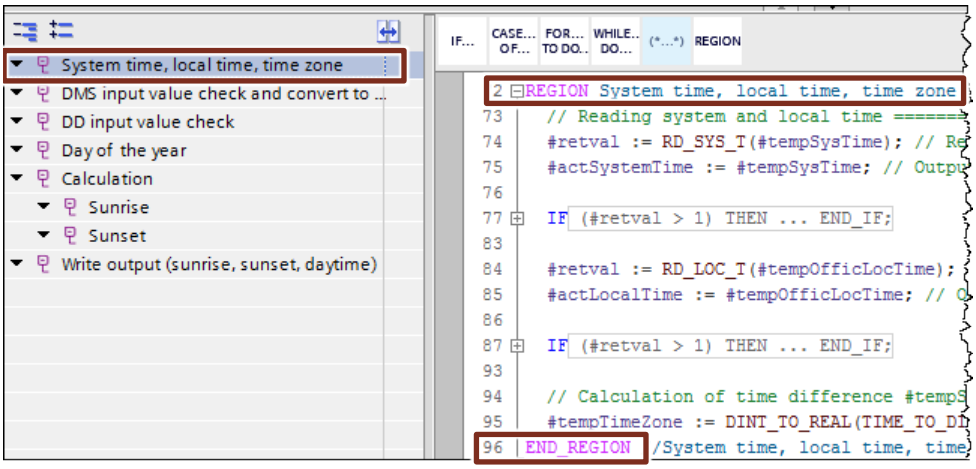
区域可以嵌套。

建议

使用关键字 REGION 来构建 SCL 块。

示例

图 3 -56：SCL 编辑器



3.10.5 正确使用 FOR、REPEAT 和 WHILE 循环

循环的使用有不同的版本和应用程序。以下示例显示了差异。

FOR 循环属性：

FOR 循环经过**定义的运行次数**。循环变量在开始时被分配一个起始值。之后，它以指定的步长递增到每个循环运行的最终值。

出于性能的原因，开始值和结束值在开始时计算一次。因此，循环变量不再影响循环代码。

语法

```
FOR statCounter : = statStartCount TO statEndCount DO
    // 语句部分;
END_FOR;
```

使用 EXIT 命令可以随时中断循环。

WHILE 循环属性：

WHILE 循环由终止条件结束。在循环代码**开始之前**检查**终止条件。**即，如果条件没有满足，则不执行循环。可以为循环代码中的下一次运行调整每个变量。

语法

```
WHILE condition DO
     // 语句部分;
END_WHILE;
```

REPEAT 循环属性：

REPEAT 循环由终止条件结束。在循环代码的**末尾**检查**终止条件。**这意味着循环**至少运行一次。**可以为循环代码中的下一次运行调整每个变量。

语法

```
REPEAT
    // 语句部分;
UNTIL condition
END_REPEAT;
```

建议

- 如果明确定义了循环变量，请使用 FOR 循环。
- 如果在循环处理期间必须调整循环变量，请使用 WHILE 或 REPEAT 循环。

3.10.6 高效地使用 CASE 指令

使用 SCL 中的 CASE 指令，它将准确地跳转到选定的 CASE 块条件。执行 CASE 块后，指令完成。例如，这使您可以更具体、更轻松地检查经常需要的值范围。

示例

```
CASE #myVar OF
        5:
                #Engine(#myParam);
        10,12:
                #Transport(#myParam);
        15:
                #Lift(#myParam);
        0..20:
                #Global(#myParam);
// 永远不会为值 5、10、12 或 15 调用 Global 函数块!
        ELSE
END_CASE;
```

注意 CASE 指令也适用于 CHAR、STRING 数据类型以及元素（参见第 2.8.5 章中的示例 2.8.5 数据类型 VARIANT（S7-1500 和 S7-1200 的 FW4.1 以上版本））。

3.10.7 不能操作循环计数器的 FOR 循环

SCL 中的 FOR 循环是纯计数器循环，即在进入循环时迭代次数是固定的。在 FOR 循环中，循环计数器不能更改。

使用 EXIT 指令可以随时中断循环。

优点

- 编译器可以更好地优化程序，因为它不知道迭代次数。

示例

```
FOR # statVar : = #statLower TO #statUpper DO
```

```
#statVar : = #statVar + 1; //无效，编译器警告
END_FOR ;
```

3.10.8 FOR 向后循环

在 SCL 中，您还可以向后或以另一个步宽递增 FOR 循环的索引。为此，请在循环头中使用可选的“BY”关键字。

例子

```
FOR # statVar : = #statUpper TO #statLower BY -2 DO

END_FOR ;
```

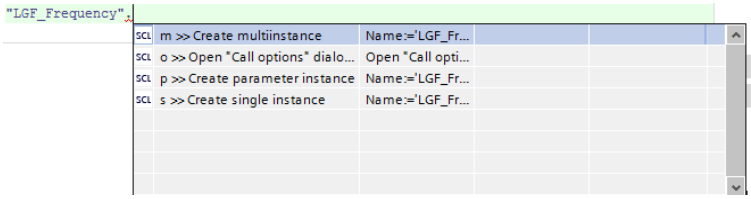
如果您将“BY”定义为“-2”，如示例中所示，则计数器在每次迭代中减 2。如果省略“BY”，则“BY”的默认设置为 1

3.10.9 轻松创建调用实例

如果您更喜欢使用键盘工作，可以很简单地在 SCL 中为块创建实例。

示例

表 3 -57：轻松创建实例

步骤	操作说明
1.	给块起一个名字，后跟一个“.”。自动编译现在向您显示以下内容。 
2.	在顶部，您可以看到已经存在的实例。此外，您可以直接创建新的单个实例或多重实例。 使用快捷方式“s”或“m”直接转到自动编译窗口中的相应条目。

3.10.10 时间变量的处理

可以像计算普通数字一样在 SCL 中计算时间变量，即不需要寻找额外的函数，例如 T_COMBINE，但可以使用简单的算术。这种方法被称为“操作数重载”。SCL 编译器会自动使用合适的函数。可以对时间类型使用合理的算法，因此可以更有效地编程。

示例

```
time difference := time stamp_1 - time stamp_2;
```

下表总结了重载的运算符及其背后的操作：

表 3 -58：SCL 的重载操作数

重载的操作数	操作
ltime + time	T_ADD LTime
ltime – time	T_SUB LTime
ltime + lint	T_ADD LTime
ltime – lint	T_SUB LTime
time + time	T_ADD Time
time - time	T_SUB Time
time + dint	T_ADD Time
time - dint	T_SUB Time
ldt + ltime	T_ADD LDT / LTime
ldt – ltime	T_SUB LDT / LTime
ldt + time	T_ADD LDT / Time
ldt – time	T_SUB LDT / Time
dtl + ltime	T_ADD DTL / LTime
dtl – ltime	T_SUB DTL / LTime
dtl + time	T_ADD DTL / Time
dtl – time	T_SUB DTL / Time
ltod + ltime	T_ADD LTOD / LTime
ltod – ltime	T_SUB LTOD / LTime
ltod + lint	T_ADD LTOD / LTime
ltod – lint	T_SUB LTOD / LTime
ltod + time	T_ADD LTOD / Time
ltod – time	T_SUB LTOD / Time
tod + time	T_ADD TOD / Time
tod – time	T_SUB TOD / Time
tod + dint	T_ADD TOD / Time
tod – dint	T_SUB TOD / Time
dt + time	T_ADD DT / Time
dt – time	T_SUB DT / Time
ldt – ldt	T_DIFF LDT
dtl – dtl	T_DIFF DTL
dt – dt	T_DIFF DT
date – date	T_DIFF DATE
ltod – ltod	T_DIFF LTOD
date + ltod	T_COMBINE DATE / LTOD
date + tod	T_COMBINE DATE/TOD

3.10.11 不必要的 IF 指令

程序员经常在 IF-THEN-ELSE 指令中思考。这经常导致程序中不必要的构造。

示例

```
IF (statOn1 = TRUE AND statOn2 = TRUE) THEN
  statMotor := TRUE;
ELSE
  statMotor := FALSE;
END_IF
```

建议

请记住，对于布尔请求，直接分配通常更有效。整个结构可以用一行编程。

示例

```
statMotor : = statOn1 AND statOn2;
```

4 独立于硬件的编程

为了确保一个块可以在所有控制器上使用而无需任何进一步的调整，重要的是不要使用硬件相关的功能和属性。

4.1 S7-300/400 和 S7-1200/1500 的数据类型

下面是所有基本数据类型和数据组的列表。

建议

- 仅使用运行程序的控制器支持的数据类型。

表 4 -1：基本数据类型符合标准 EN 61131-3

	描述	S7-300/400	S7-1200	S7-1500
位数据类型	• BOOL • BYTE • WORD • DWORD	是	是	是
	• LWORD	否	否	是
字符类型	• CHAR（8 位）	是	是	是
数值数据类型	• INT（16 位） • DINT（32 位） • REAL（32 位）	是	是	是
	• SINT（8 位） • USINT（8 位） • UINT（16 位） • UDINT（32 位） • LREAL（64 位）	否	是	是
	• LINT（64 位） • ULINT（64 位）	否	否	是
时间类型	• TIME • DATE • TIME_OF_DAY	是	是	是
	• S5TIME	是	否	是
	• LTIME • L_TIME_OF_DAY	否	否	是

表 4 -2：由其他数据类型组成的数据组

	描述	S7-300/400	S7-1200	S7-1500
时间类型	• DT (DATE_AND_TIME)	是	否	是
	• DTL	否	是	是
	• LDT (L_DATE_AND_TIME)	否	否	是
字符类型	• STRING	是	是	是
数组	• ARRAY	是	是	是
结构	• STRUCT	是	是	是

表 4 -3：在块之间传输的形式参数的参数类型

	描述	S7-300/400	S7-1200	S7-1500
指针	• POINTER • ANY	是	否	是 [1)]
	• VARIANT	否	是	是
块	• TIMER • COUNTER	是	是 [2)]	是
	• BLOCK_FB • BLOCK_FC	是	否	是
	• BLOCK_DB • BLOCK_SDB	是	否	否
	• VOID	是	是	是
PLC 数据类型	• PLC 数据类型	是	是	是

1) 为了优化访问，只能使用符号寻址

2) 对于 S7-1200/1500，TIMER 和 COUNTER 数据类型由 IEC_TIMER 和 IEC_Counter 表示。

4.2 不使用位存储器而使用全局数据块

优点

- 优化的全局 DB 显然比仅出于兼容性原因而未优化的位存储器地址区域更强大。

建议

- 使用位存储器处理（包括系统和时钟标志）是有问题的，因为每个控制器的位存储器区域的大小是不同的。不要使用位存储器进行编程，而是始终使用全局数据块。这才是程序可以普遍使用的方式。

4.3 “循环位”编程

建议

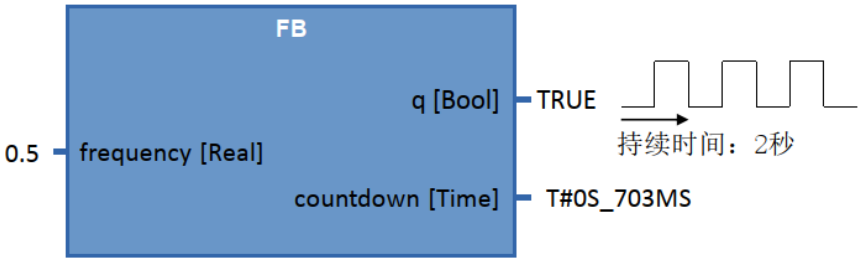
对于时钟存储器的编程，硬件配置必须始终正确。

使用编程实现时钟发生器的程序块。您可以在下面看到一个使用 SCL 编程语言的时钟发生器的编程示例。

示例

程序块具有以下功能。预设频率。“q”输出一个根据所需频率切换的布尔值。“countdown”输出 “q”当前状态的剩余时间。

如果所需频率小于或等 0.0，则 q = FALSE 并且 countdown = 0.0。



注意

完整的编程示例可以在以下条目中找到：

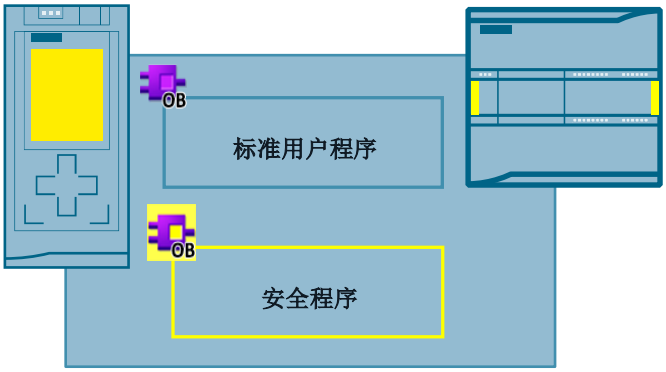
https://support.industry.siemens.com/cs/ww/en/view/109479728

5 TIA 博途中的 STEP 7 Safety

5.1 介绍

TIA 博途 V13 SP1 或更高版本支持故障安全 S7-1200F 和 S7-1500F CPU。在这些控制器中，标准和故障安全编程可以在一个设备中进行。使用 SIMATIC STEP 7 Safety（TIA 博途）选件包对故障安全用户程序进行编程，。

图 5 -1：标准和安全程序



优点

- 使用工程工具：TIA 博途，对标准和故障安全程序中进行统一编程
- 熟悉的编程语言 LAD 和 FBD
- 统一的诊断和在线功能

注意 故障安全并不意味着程序不包含错误。程序员负责正确的编程逻辑。

故障安全意味着确保控制器中的故障安全用户程序正确处理。

注意 有关安全主题的更多信息，例如安全要求或安全程序原理，请参见：

TIA 博途 – 最重要文档和链接的概述 – 安全
https://support.industry.siemens.com/cs/ww/en/view/90939626

应用程序和工具 – Safety Integrated
https://support.industry.siemens.com/cs/ww/en/ps/14675/ae

STEP 7 Safety (TIA 博途) – 手册
https://support.industry.siemens.com/cs/ww/en/ps/14675/man

5.2 术语

本文档始终使用具有以下含义的术语。

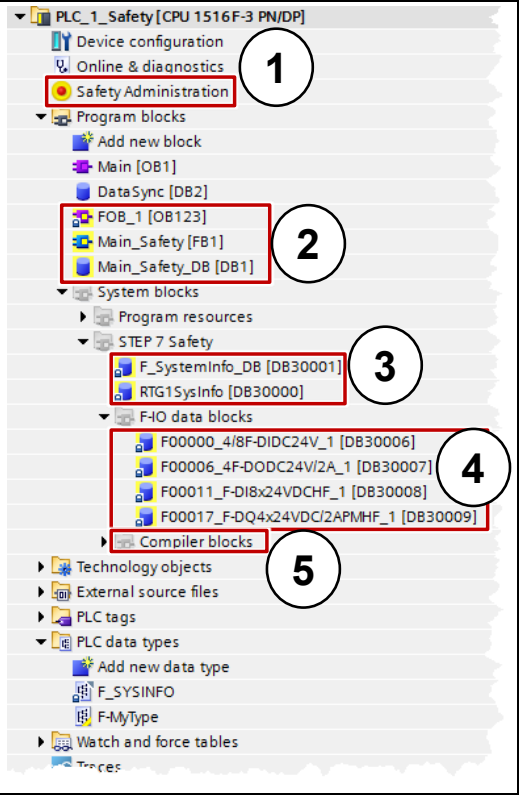
表 5 -2：安全术语

术语	描述
标准用户程序	标准用户程序为与 F 编程无关的程序部分。
安全程序 （F 程序、故障安全用户程序）	故障安全用户程序是控制器独立进行故障安全处理的程序部分。 为了区分标准用户程序的块和指令，所有故障安全块和指令在软件用户界面（例如在项目导航中）都带有黄色阴影。在硬件配置中 F-CPU 和 F-I/O 的故障安全参数以黄色阴影显示。

5.3 安全程序的组成部分

这个安全程序始终由用户编写、系统生成 F 块和“安全管理”编辑器组成。

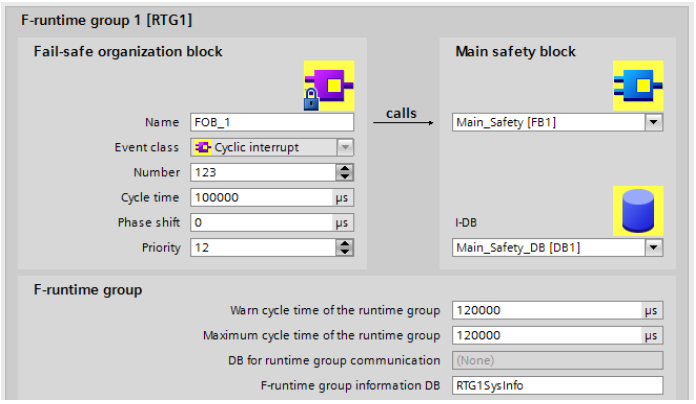
表 5 -3：安全程序的组成部分

描述	界面
1. “安全管理”编辑器 - 安全程序的状态 - 集体 F 签名 - 安全操作状态 - 创建/组织 F 运行组 - 关于 F 块的信息 - 关于 F 型 PLC 数据类型的信息 - 定义/更改访问保护 2. 用户创建的 F 块 3. 系统生成的 F 运行块 - 块包含有关 F 运行组的状态信息 4. 系统生成的 F-I/O 数据块 - 块包含用于评估 F 模块的变量 5. “编译器块” 系统生成的验证块 - 它们在控制器的后台运行，并提供安全程序的故障安全处理 - 用户无法处理这些块	

5.4 F-运行组

安全程序始终在具有定义周期的 F 运行组中处理。F 运行组由一个“故障-安全组织块”组成，该块称为“主安全块”。所有用户编写的安全功能块被“主安全块”调用。

图 5 -4：5



优点

- 运行组可以在“安全管理”中简单地创建和配置。
- 运行组中的 F 块是自动创建的。

特性

- 最多可以创建两个 F 运行组。

5.5 F 签名

每个 F 组件（站、I/O、块）都有唯一的 F 签名。使用 F 签名可以快速检测 F 设备配置、F 块或完整站是否仍与原始配置或编程一致。

优点

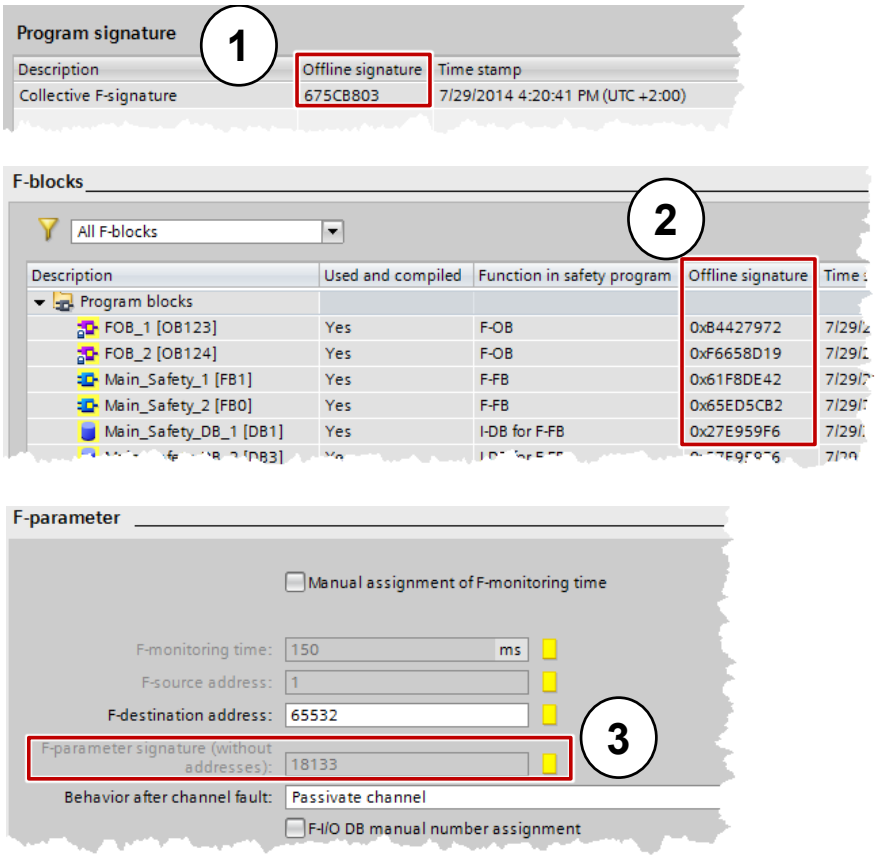
- 简单快速地比较 F 块和 F 设备配置

特性

- F 参数签名（没有 F-I/O 的地址）…
  - 只能通过调整参数来改变。
  - PROFIsafe 地址改变时保持不变。但是，该站的集体 F 签名发生了变化。
- 只有当 F 块中的逻辑发生变化时，F 块签名才会改变。
- 更改以下内容 F 块签名保持不变
  - 块编号
  - 块接口
  - 块版本

示例

图 5 -6：F 签名示例



1. “安全管理”编辑器中的站的 F 集体签名
2. “安全管理”编辑器中的 F 块签名（也可以从块的属性中读取）
3. “设备和网络”中“设备视图”中的 F 参数签名

注意 对于 S7-1500F 控制器，可以直接在安装的显示屏或集成的 Web 服务器上读取所有 F 签名。

5.6 在 F-I/O 上分配 PROFIsafe 地址

每个 F-I/O 设备都有一个 PROFIsafe 地址，用于识别和与 F 控制器通信。分配 PROFIsafe 地址时，可以进行两种不同的配置。

表 5 -7：设置 F 地址

ET 200M / ET 200S （ PROFIsafe 地址类型 1）	ET 200MP / ET 200SP （ PROFIsafe 地址类型 2）
通过模块的拨码开关直接在分配 PROFIsafe 地址 TIA 博途的设备配置与拨码开关位置的 PROFIsafe 地址必须相同。	通过 TIA 博途分配 PROFIsafe 地址 配置的 PROFIsafe 地址被加载到模 块的智能编码元件上。

优点

- 更换 F 模块无需重新分配 ET 200MP 和 ET 200SP 的 PROFIsafe 地址。在模块更换期间，智能编码元件保留在基座中。
- 由于 TIA 博途指示 PROFIsafe 地址警告分配错误，因此配置简单。
- 一个 ET 200SP 的所有 F 模块的 PROFIsafe 地址可以同时分配。

注意

关于为 F-I/O 分配 PROFIsafe 地址的更多信息，请访问：

SIMATIC 工业软件 SIMATIC 安全 – 组态和编程
https://support.industry.siemens.com/cs/ww/en/view/54110126

5.7 F-I/O 评估

F-I/O 的所有当前状态都保存在 F-I/O 块中。在安全程序中，状态可以被评估和处理。 S7-1200F/1500F 和 S7-300F/400F 之间存在以下差异。

表 5 -8：S7-300F/400F 和 S7-1500F 的 F-I/O DB 中的变量

F-I/O DB 中的变量或 IO 中的值状态	S7 -300/400F 的 F-I/O	S7 -1200F/1500F 的 F-I/O
ACK_NEC	是	是
QBAD	是	是
PASS_OUT	是	是
QBAD_I_xx *	是	否
QBAD_O_xx *	是	否
值状态	否	是

* QBAD_I_xx 和 QBAD_O_xx 显示通道值的有效性，并对应于 S7-1200F/1500F 的值状态取反（更多信息见下一章节）。

5.8 值状态(S7-1200F/1500F)

除了诊断消息以及状态和错误显示外，F 模块还提供有关每个输入和输出信号有效性的信息 – 值状态。值状态的存储方式与过程映像中输入信号的存储方式相同：

值状态通知相应通道值的有效性。

- 1：通道输出一个有效的过程值。
- 0：通道输出一个替代值。

表 5 -9：Q_BAD (S7-300F/400F)和值状态(S7-1200F/1500F)的区别

使用场景	QBAD(S7-300F/400F)	值状态 (S7-1200F/1500F)
F-I/O 上的有效值（无错误）	FALSE	TRUE
发生通道错误	TRUE	FALSE
通道错误离去(ACK_REQ)	TRUE	FALSE
确认故障(ACK_REI)	FALSE	TRUE

特性

- 值状态被附加到输入和输出模块的输入过程映像中。
- 一个 F-I/O 的通道值和值状态只能被同一个 F 运行组访问。

建议

- 为提高可读性，将结尾设定为“VS”，例如“TagIn1VS”作为值状态的符号名称。

示例

以 F-DI 8x24VDC HF 模块为例，过程映像中值状态位的位置。

表 5 -10：过程映像中的值状态位，以 F-DI 8x24VDC HF 为例

F-CPU 中的字节	F-CPU 中的分配位							
	7	6	5	4	3	2	1	0
x + 0	DI_7	DI_6	DI_5	DI_4	DI_3	DI_2	DI_1	DI_0
x +1	DI_7 的值状态	DI_6 的值状态	DI_5 的值状态	DI_4 的值状态	DI_3 的值状态	DI_2 的值状态	DI_1 的值状态	DI_0 的值状态

x = 模块起始地址

注意

有关所有 ET 200SP 模块的值状态的更多信息，请访问：

故障安全 CPU – 手册
https://support.industry.siemens.com/cs/ww/en/ps/13719/man

故障安全 I/O 模块 – 手册
https://support.industry.siemens.com/cs/ww/en/ps/14059/man

5.9 数据类型

5.9.1 概述

S7-1200/1500F 的安全程序的不受限制的数据类型范围。

表 5 -11：安全数据类型

类型	大小	数值范围
BOOL	1 位	0 .. 1
INT	16 位	-32.768 .. 32.767
WORD	16 位	-32.768 .. 65.535
DINT	32 位	-2.14 .. 2.14 Mio
TIME	32 位	T#-24d20h31m23s648ms 到 T#+24d20h31m23s647ms

5.9.2 隐式转换

在与安全相关的应用中，可能需要使用不同数据类型的变量执行数学函数。为此所需的功能块需要形式参数来定义数据格式。如果操作数不符合预期的数据类型，则必须先进行转换。

在以下情况下，S7-1200/1500 也可以隐式执行数据转换：

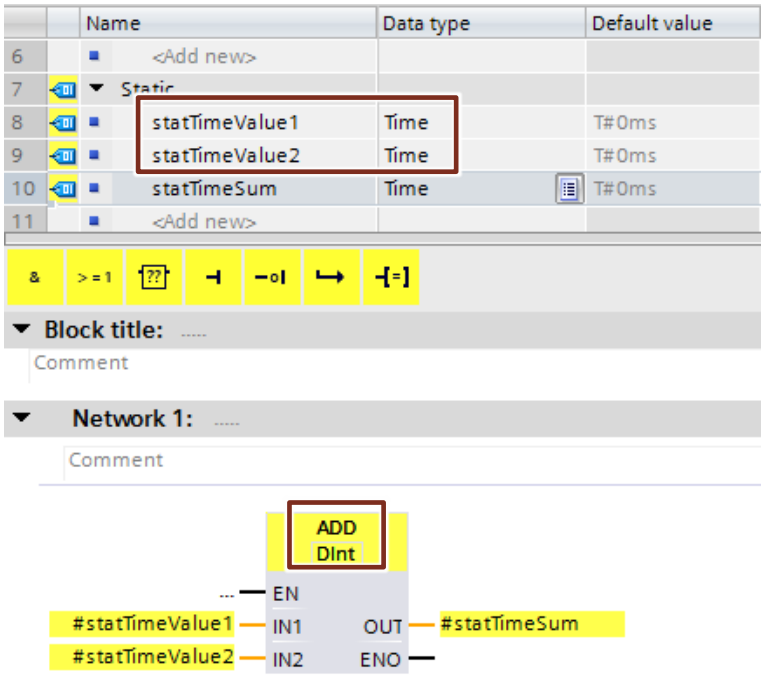
- IEC 检查被禁用。
- 数据类型具有相同的长度。

因此，以下数据类型可以在安全程序中隐式转换：

- WORD ↔ INT
- DINT ↔ TIME

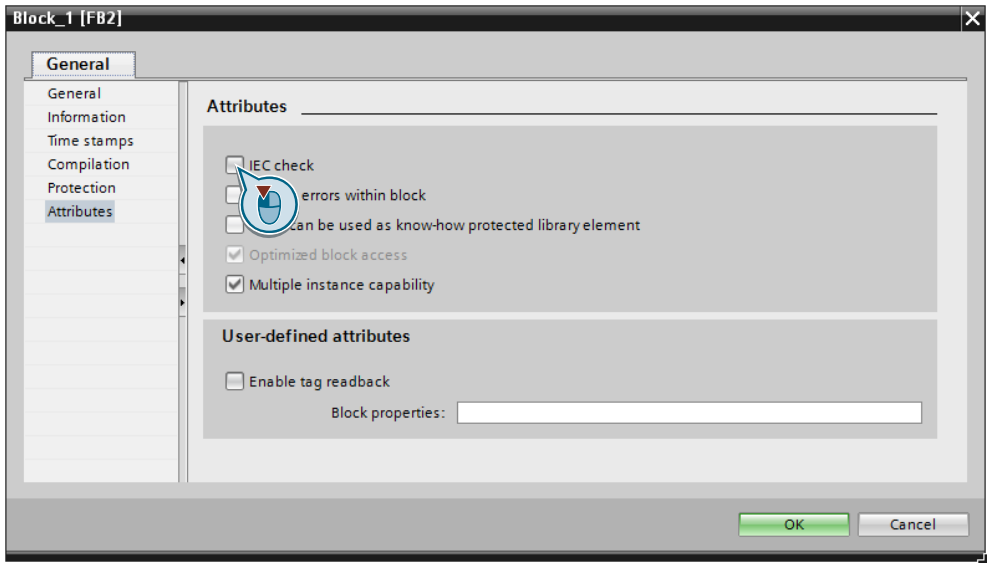
一个实际的应用是两个时间值的相加，尽管函数“Add”需要作为“DInt”输入。结果也作为“时间”变量输出。

图 5 -12：两个时间值相加



在相应函数块或函数的属性中启用或禁用 IEC 检查。

图 5 -13：禁用 IEC 检查



5.10 F-compliant 型 PLC 数据类型

对于安全程序，也可以使用 PLC 数据类型优化数据结构。

优点

- PLC 数据类型的更改会自动更新到用户程序的所有使用位置。

特性

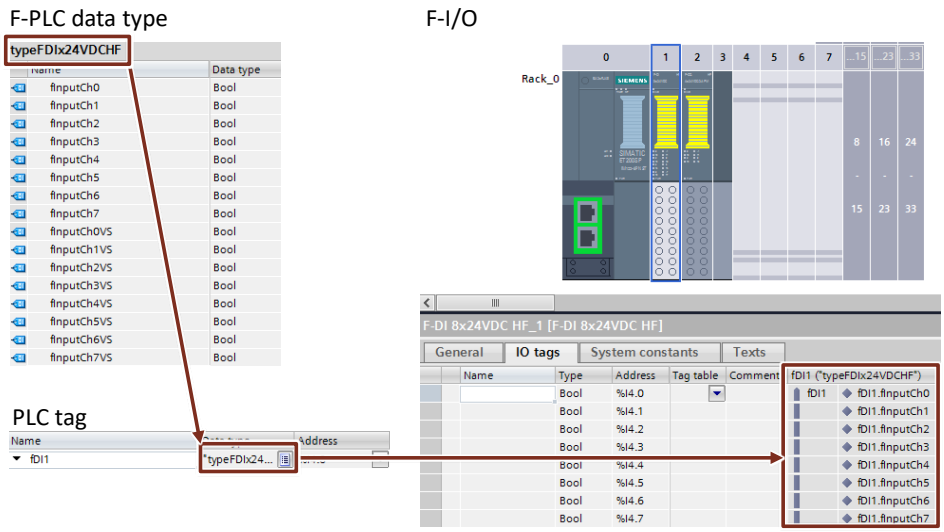
- F-PLC 数据类型的声明和使用方式与 PLC 数据类型相同。
- 作为 F-PLC 数据类型，安全程序中允许的所有数据类型都可以使用。
- 不支持 F-PLC 数据类型在其他 F-PLC 数据类型中嵌套。
- F-PLC 数据类型既用于安全程序，也可用于标准用户程序。

建议

- 使用 F-PLC 数据类型访问 I/O 区域（见章节 3.6.5 访问具有 PLC 数据类型的 I/O 区域）
- 这里必须遵守以下规则：
  - F PLC 数据类型的变量结构必须与 F-I/O 的通道结构相匹配。
  - 例如，符合 F 的 PLC 数据类型，具有 8 个通道的 F-I/O：
    - 8 个 BOOL 变量（通道值）或
    - 16 个 BOOL 变量（通道值 + 值状态）
  - 只有激活的通道才允许访问 F-I/O。当组态 1oo2（2v2）评估时，较高的通道始终是失效的。

示例

图 5 -614：使用 F-PLC 数据类型访问 I/O 区域



5.11 TRUE/FALSE

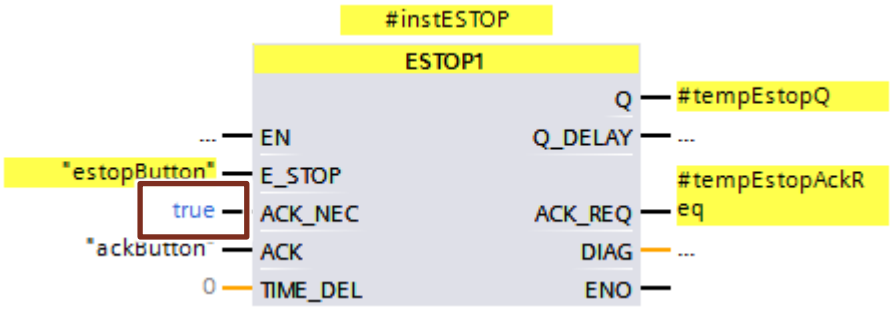
安全程序中“TRUE”和“FALSE”信号的使用可分为两种应用情况：

- 作为块的实际参数
- 作为对操作的分配

块的实际参数

对于 S7-1200F/1500F 控制器，您可以使用布尔常量“FALSE”表示 0，“TRUE”表示 1 作为实际参数，以便在安全程序中的块调用期间提供给形式参数。只有关键字“FALSE”或“TRUE”被写入形式参数。

图 5 -15： “TRUE”和“FALSE”信号作为实际参数



操作分配

要为操作创建“TRUE”或“FALSE”信号，请执行以下操作：

1. 创建两个 BOOL 类型的静态变量“statTrue”和“statFalse”。
2. 将默认值“false”分配给“statFalse”变量。
3. 将默认值“true”分配给“statTrue”变量。

您可以在完整的功能块中使用变量作为“True”和“False”读取信号。

图 5 -16： “TRUE”和“FALSE”信号

	Name	Data type	Default value	Retain
	Static			
	statTrue	Bool	true	Non-retain
	statFalse	Bool	false	Non-retain

5.12 优化编译和程序运行

安全程序的一个重要部分是通过编码处理对用户编程进行保护。目的是发现安全程序中的任何类型的数据损坏，从而防止出现不安全的情况。

该保护程序是在编译期间创建的，因此会延长编译时间。F-CPU 的运行时间也会因保护程序而延长，因为 F-CPU 会进行将结果与用户程序进行比较的额外处理。

系统自动生成的保护程序可以在 F-CPU 的系统块文件夹中找到。

示例

图 5 -17：用户和系统创建的 F 块

本章展示了缩短编译和程序运行的不同选项。

根据使用情况，并非要听取所有的使用建议。然而，它们提供了一些信息，说明为什么某些编程方法会比非优化程序有更短的编译和程序运行。

5.12.1 避免时间处理块：TP、TON、TOF

每个时间处理块（TP、TON、TOF）都需要保护代码中的附加块和全局数据更正。

建议

尽可能少地使用这些块。

5.12.2 避免深层调用层次结构

深层调用层次结构扩大了系统创建的 F 块的代码，因为需要更大范围的保护功能和测试。当嵌套深度超过 8 时，TIA 博途将在编译期间发出警告。

建议

构建的程序避免不必要的深层调用层次结构的方式。

5.12.3 避免 JMP/LABEL 结构

如果通过 JMP/LABEL 跳转块调用，则会在系统侧的 F 块中产生额外的保护。这里，必须对跳过的块调用执行校正码。这会在编译中消耗性能和时间

建议

尽量避免 JMP/LABEL 结构，以减少系统侧的 F-block。

5.13 标准程序与 F 程序之间的数据交换

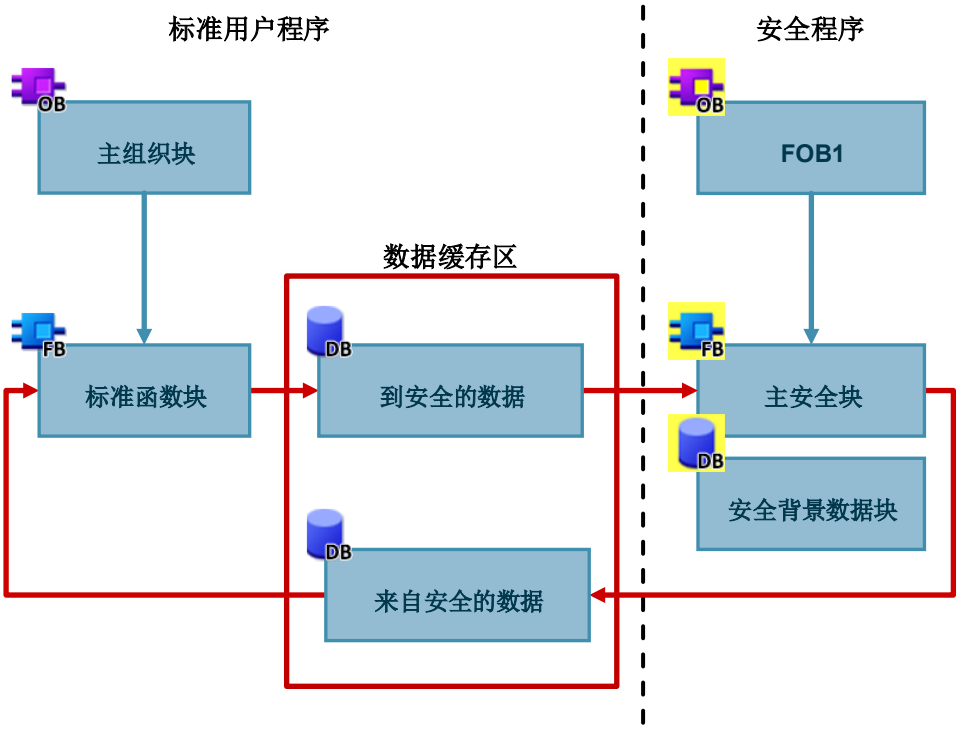
在某些情况下，需要在安全程序和标准用户程序之间交换数据。为了保证标准和安全程序之间的数据一致性，应格外注意以下建议。

建议

- 不通过位存储器进行数据交换（参见章节 4.2 不使用位存储器而使用全局数据块）
- 将安全程序和标准用户程序之间的访问集中在两个标准 DB 上。

  因此，标准程序的更改不会影响安全程序。控制器也不需要处于 STOP 模式来加载标准程序。

图 5 -18：标准和安全程序之间的数据交换



5.14 测试安全程序

除了标准用户程序的始终可控数据之外，还可以在禁用的安全模式下更改安全程序的以下数据。

- F-I/O 的过程映像
- F-DB（F-运行组通信的 DB 除外）， F-FB 的背景 DB
- F-I/O DB

特性

- 只有在 F -CPU RUN 模式下才能控制 F I/O。
- 从监视表中可以控制在一个安全程序中的最多 5 个输入/输出。
- 可以使用多个监视表。
- 作为触发点，需要为“循环开始”或“循环结束”设置“永久”或“一次”。
- F-I/O 不能强制。
- 如果仍希望使用断点进行测试，则需要事先停用安全模式。这会导致以下错误：
  - 与 F-I/O 通信时出错
  - 故障安全 CPU-CPU 通信错误

5.15 发生 F 错误时的 STOP 模式

在以下情况下，会触发 F-CPU 的 STOP 模式：

- 不得在“系统块”文件夹中添加、更改或删除任何块。
- 不得对未在安全程序中调用的 F-FB 的背景 DB 进行任何访问。
- 不得超过“F 运行组的最大循环时间”。在 F 运行组的两次调用之间最多可以选择为“F 运行组的最大循环时间”（最大 20000 ms）。
- 如果从未运行的运行组的 F 运行组通信的 DB 中读取变量（未调用 F 运行组的主安全块）。
- 不允许在线和离线编辑 F-FB 的背景 DB 中的起始值，这会导致 F-CPU 停止。
- 主安全块不能包含任何参数，因为它们无法提供。
- F-FC 的输出必须始终进行初始化。

5.16 安全程序的移植

有关移植安全程序的信息，请访问：

https://support.industry.siemens.com/cs/ww/en/view/109475826

5.17 有关安全的常规建议

通常，以下建议适用于处理 STEP 7 Safety 和 F 模块。

- 只要有可能，总是使用 F 控制器。因此，可以很容易地实现安全功能的后期扩展。
- 始终使用一个密码的安全程序，以防止未经授权的更改。密码在“安全管理”编辑器中设置。

6 使用用户程序自动生成可视化

6.1 介绍

自 TIA 博途 V14 起，可以使用 SiVArc（SIMATIC Visualization Architect）选件包从可视化库和控制器中的用户程序自动生成工厂的可视化。

在本章中，为与 SiVArc 一起使用，将优化用户程序。

使用 SiVArc 生成的优势

- 具有过程连接的可视化自动生成
- 用户界面标准化
- 对操作屏幕进行简单和一致的调整

要求

使用 SiVArc 的基本要求是工厂的**高度标准化**。将系统模块化为单独的功能组的优势在于，SiVArc 可以使用这些功能组从现有的画面库生成操作画面并将其互连。接口的标准化有助于高效工作和可视化的自动生成。

遵守本编程指南的常规建议将有所帮助。

注意

SiVArc 是 HMI 和控制器之间的接口主题。本节将从控制端介绍 SiVArc。

以下链接将让您深入了解 SiVArc 的功能：

应用示例“SiVArc Getting Started”
https://support.industry.siemens.com/cs/ww/en/view/109740350

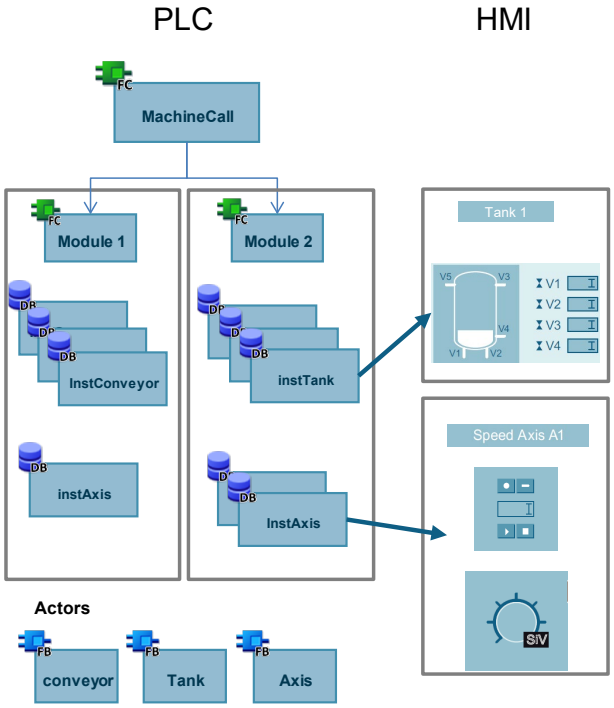
SiVArc 手册
https://support.industry.siemens.com/cs/ww/en/view/109755214

SITRAIN 课程：SIMATIC 可视化架构，自动化 HMI 生成
https://support.industry.siemens.com/cs/ww/en/view/109758628

6.2 自动生成的工作原理

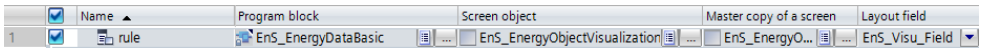
在用户程序中调用标准化块，例如用于发动机控制的块。使用所谓的 SiVArc 规则，可以将调用的块与可视化元素（文本字段、IO 域、图像块等）链接起来。

图 6 -1：控制系统中的调用层次结构示例



SiVArc 使用规则为每个指定块的调用在图像模板的副本上生成指定的可视化元素。

图 6 -2：SiVArc 规则



注意 还可以选择在控制器中限制或阻止规则的执行。

6.3 控制 HMI 生成器

用于控制 HMI 生成器，有以下选项：

- 禁用特定调用的 SiVArc 生成
- 根据功能或工厂位置对 SiVArc 生成进行排序
- 为 SiVArc 生成添加更多属性

在 SiVArc 规则中，您可以使用来自控制器的以下信息：

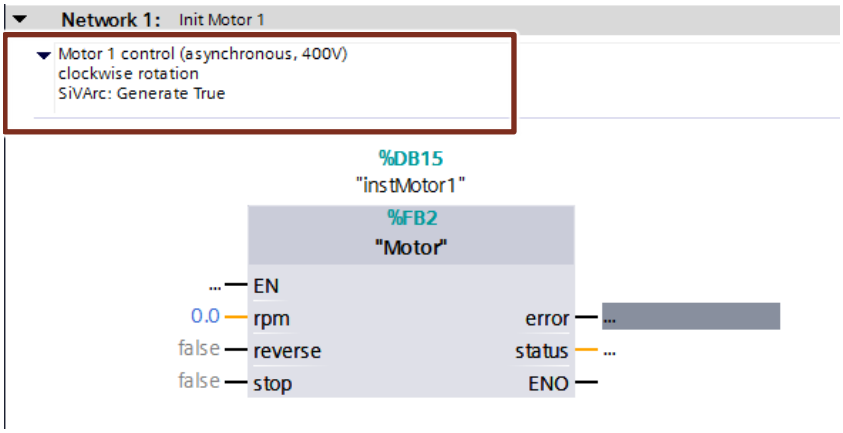
- 网络注释
- SiVArc 变量

这提供了在控制器中为 SiVArc 提供附加信息的选项，可以将其用作 SiVArc 规则中的条件。

6.3.1 使用网络注释进行控制

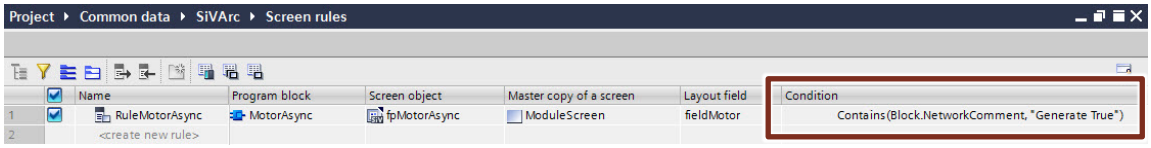
在网络注释中，您可以添加有关控制 SiVArc 规则在生成期间搜索的生成器的信息：

图 6 -3：SiVArc 扩展的网络注释



在规则编辑器中，使用“条件”列中的“包含”功能并搜索信息，例如“包含（Block.NetworkComment，“字符串”）”。

图 6 -4：在规则中使用网络注释



使用此功能，就有了一个自由的设计框架来限制规则的执行，或仅对某些网络执行规则。

建议

在网络注释中清楚地标记 SiVArc 生成的信息，例如“SiVArc :Generate True”。

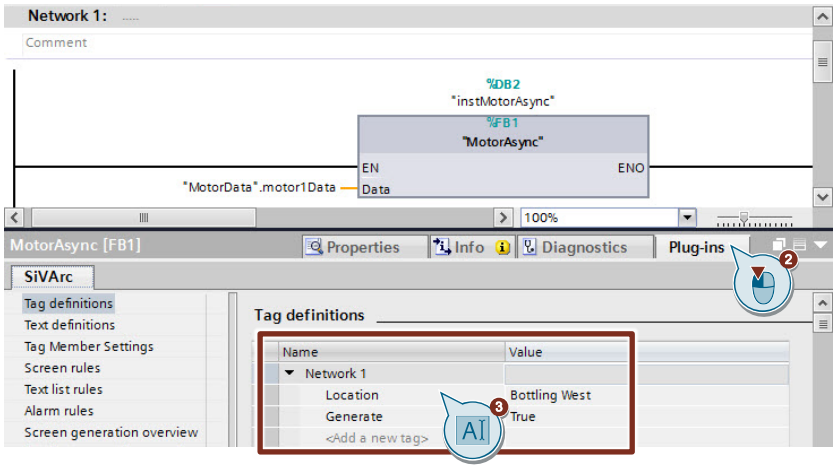
6.3.2 使用 SiVArc 变量进行控制

可以为设备中的每个网络定义特殊的 SiVArc 变量，并在 SiVArc 规则中使用这些变量。

要创建 SiVArc 标签，请执行以下操作：

1. 打开块。
2. 切换到检查窗口中的“插件”选项卡。
3. 对于每个网络，在“名称”列中输入变量的名称，在“值”列中输入字符串类型的值。

图 6 -5：创建 SiVArc 变量

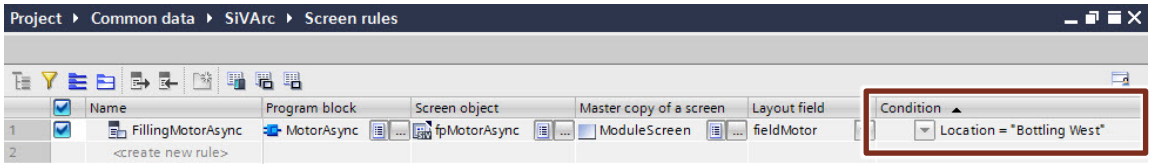


在规则编辑器中，您可以使用表达式“变量名称” = “值”查询 SiVArc 变量，从而影响规则的执行。

在规则编辑器中，在“条件”列中输入 SiVArc 变量，例如“Location” = “Bottling West”。

在本例中，只有当 SiVArc 变量“Location”在要生成的网络中被设置为值“Bottling West”时，才会执行该规则。

图 6 -6：在规则中使用 SiVArc 变量



建议

使用通用变量名称来控制生成器以简化规则创建。
下表显示了 SiVArc 变量的示例：

表 6-1： SiVArc 变量示例

名称	值	含义
Generate	true,false	SiVArc 为该网络生成元素。
Location	Bottling West，Mixing 等。	可以使用此功能自动将元素分配给 HMI 图像。

注意 必须使用规则编辑器中的条件自己定义“含义”列中列出的效果。SiVArc 变量本身对生成器没有影响。

6.4 附加建议

仅使用 WinCC 支持的字符

仅使用 WinCC 支持的字符来指定变量。

生成图像时， SiVArc 访问数据块、变量或网络注释的标识符。SiVArc 删除 WinCC 不支持的所有字符。
这会导致项目的不一致。

不支持的字符：

- %,@, ?, ", /, \, <, >, ., :

使用编程语言 FBD 调用块

为了可以通过网络注释或 SiVArc 变量控制生成器，请在编程语言 FBD 中调用要生成可视化元素的块。

注意 此建议适用于并包括 TIA 博途 V15。

自 TIA 博途 V15.1 起，也可以使用 SCL 模块。

7 最重要建议

- 使用优化块
  - 第 2.6 章 优化块
- 使用数据类型 VARIANT 而不是 ANY
  - 第 2.8.5 章 数据类型 VARIANT
- 清晰而良好地构建程序
  - 第 3.2 章 程序块
- 插入指令作为多重实例（TON，TOF .. ）
  - 章节 3.2.5 多重实例
- 块的可重用编程
  - 第 3.2.9 章 块的可重用性
- 符号编程
  - 第 3.6 章 符号寻址
- 处理数据时，使用 ARRAY
  - 第 3.6.2 章 ARRAY 数据类型和间接寻址访问
- 创建 PLC 数据类型
  - 第 3.6.5 章 访问具有 PLC 数据类型的 I/O 区域
- 使用库来存储程序元素
  - 第 3.7 章 库
- 不使用位存储而使用全局数据块
  - 第 4.2 章 不使用位存储器而使用全局数据块

8 附录

8.1 服务和支持

工业在线支持

您有任何问题或需要帮助吗？

西门子工业在线支持提供全天候访问我们的全部服务和支持知识和产品组合。

工业在线支持是有关我们产品、解决方案和服务的信息的中心地址。

产品信息、手册、下载、常见问题解答、应用示例和视频 – 只需单击几下鼠标即可访问所有信息：https://support.industry.siemens.com

技术支持

西门子工业的技术支持为您提供有关所有技术查询的快速和有能力的支持，并提供大量定制服务——从基本支持到个人支持合同。请通过 Web 表单向技术支持发送查询：
www.siemens.com/industry/supportrequest

SITRAIN – 工业培训

我们为您提供全球可用的工业培训课程，提供实践经验、创新的学习方法和针对客户特定需求量身定制的理念。

有关我们提供的培训和课程及其地点和日期的更多信息，请访问我们的网页：
www.siemens.com/sitrain

服务提供

我们的服务范围包括：

- 工厂数据服务
- 备件服务
- 维修服务
- 现场和维护服务
- 改造和现代化服务
- 服务计划和合同

可以在服务目录网页中找到有关我们服务范围的详细信息：

https://support.industry.siemens.com/cs/sc

工业在线支持应用程序

无论您身在何处，通过“西门子工业在线支持”应用程序您都将获得最佳支持。该应用程序适用于 Apple iOS、 Android 和 Windows Phone：

https://support.industry.siemens.com/cs/ww/en/sc/2067

8.2 链接和文献

表 8 -1：链接和文献

	话题
\1\	西门子工业在线支持 https://support.industry.siemens.com
\2\	下载页面的进入 https://support.industry.siemens.com/cs/ww/en/view/81318674
\3\	S7-1200 和 S7-1500 的编程风格指南 https://support.industry.siemens.com/cs/ww/en/view/81318674
\4\	用于 STEP 7(TIA 博途)和 S7-1200/S7 -1500 的(LGF)通用函数库 https://support.industry.siemens.com/cs/ww/en/view/109479728
\5\	STEP 7(TIA 博途)和 S7-1200/S7 -1500 的 PLC 数据类型(LPD)库 https://support.industry.siemens.com/cs/ww/en/view/109482396
\6\	TIA 博途- 最重要文档和链接的概述 https://support.industry.siemens.com/cs/ww/en/view/65601780
\7\	STEP 7(TIA 博途)手册 https://support.industry.siemens.com/cs/ww/en/ps/14673/man
\8\	S7-1200 (F)手册 https://support.industry.siemens.com/cs/ww/en/ps/13683/man
\9\	S7-1500 (F)手册 https://support.industry.siemens.com/cs/ww/en/ps/13716/man
\10\	ET 200SP CP 手册 https://support.industry.siemens.com/cs/ww/en/ps/13888/man
\11\	S7-1200 入门 https://support.industry.siemens.com/cs/ww/en/view/39644875
\12\	S7-1500 入门 https://support.industry.siemens.com/cs/ww/en/view/78027451
\13\	SIMATIC S7-1200 / S7-1500 基于国际助记符的编程语言对照表 https://support.industry.siemens.com/cs/ww/en/view/86630375

8.3 文档更改

表 8 -2：文档更改

版本	日期	修改
V1.0	09/2013	第一版
V1.1	10/2013	以下章节中的更正： 2.6.3 适用于 S7-1500 的处理器优化数据存储 2.12 用户常量 3.2.2 函数(FC) 3.2.3 函数块(FB) 3.4.3 本地内存

版本	日期	修改
V1.2	03/2014	新章节： 2.6.4 优化和非优化变量之间的转换 2.6.6 与优化数据通信 2.9.1 MOVE 指令 2.9.2 VARIANT 指令 3.6.5 访问具有 PLC 数据类型的 I/O 区域 以下章节中的更正： 2.2 术语 2.3 编程语言 2.6 优化块 2.10 符号和注释 3.2 程序块 3.5 保持性 4.3 “循环位”编程 不同章节中的各种更正
V1.3	09/2014	新章节： 2.8.4 Unicode 数据类型 2.10.2 监视表中的注释行 2.12 用户常量 3.2.10 块的自动编号 5 TIA 博途中的 STEP 7 Safety 以下章节中的更正： 2.7 块属性 2.8 S7-1200/1500 的新数据类型 2.9 指令 2.10 符号和注释 3.6.4 STRUCT 数据类型和 PLC 数据类型 3.7 库 不同章节中的各种更正

版本	日期	修改
V1.4	11/2015	新章节： 2.6.5 优化和非优化访问的块之间的参数传输 3.3.3 参数传输概述 3.10.5 正确使用 FOR、REPEAT 和 WHILE 循环 5.12 优化编译和程序运行
V1.5	03/2017	新章节： 2.7.3 块接口 – 隐藏块参数（V14 或更高版本） 2.9.4 PLC 数据类型的变量比较（V14 或更高版本） 2.9.5 多重赋值（V14 或更高版本） 3.2.6 作为参数的实例传递 (V14) 3.6.3 形参 Array[*](V14 或高版本) 3.6.7 LAD 和 FBD 中的 SCL 网络（V14 及更高版本） 3.10.4 使用关键字 REGION（V14 或更高版本）进行结构化 3.10.11 不必要的 IF 指令 不同章节的若干更正
V1.6	12/2018	新章节： 6 使用用户程序自动生成可视化 更新扉页和法律信息