

SIEMENS
Ingenuity for life



S7-1200/1500 编程指南

TIA Portal

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

Siemens
Industry
Online
Support



法律声明

应用示例的使用

应用示例通过几个组件的交互，来说明自动化任务解决方案，包括文本、图形和/或软件模块形式。这些应用示例都是西门子股份公司和/或西门子股份公司的子公司（“西门子”）提供的免费服务。无任何约束力，对有关配置和设备的完整性或功能性也不作要求。这些应用示例仅有助于有关典型任务的理解，不构成特定于客户的解决方案。用户自己有责任根据适用的法规，正确、安全地使用产品，检查相应应用示例的功能，并根据用户自己的系统对其进行自定义。

西门子授予用户的非排他性、不可再许可、不可转让的权利，应用示例可由经过技术培训的人员使用。对应用示例的任何更改所造成的后果都由用户负责。仅与用户自己的产品结合使用时，才允许与第三方共享应用示例或复制应用示例或其中一部分。由于无需对应用示例进行常规测试和付费产品的质量检查，可能具有功能和性能缺陷以及错误。用户有责任以不会造成任何财产损失或人身伤害的方式加以使用。

责任免除

西门子不因任何法律原因承担任何责任，包括但不限于对应用示例的使用、可用性、完整性和自由性以及相关信息、组态和性能数据以及由此造成的任何损害承担责任。该责任免除不适用于强制性责任的状况，如根据《德国产品责任法》规定的故意触犯、重大过失，或者对生命、人身或健康造成危害，对产品质量的、对产品缺陷的欺瞒或影响协约确立的违规。而且，除非是由于故意或者重大过失或者基于死亡、人身和健康损害等而发生的强制性责任，因违反了涉及合同根基的事项而导致合同根本违约而提出的索赔应当限制在合同本身可遇见的损害范围内。上述条款并不意味着改动用户对损失进行举证的责任。在此方面，除非西门子有强制责任，否则用户应就西门子对第三方的现有或将来的索赔要求承担责任。

在使用这些应用示例时，用户应该知道，西门子不会对超出所述责任条款范围的任何损害承担任何责任。

其它信息

西门子保留随时对这些应用示例进行更改的权利，恕不另行通知。本文档中的建议条款和西门子的其它出版物（例如样本）存在不一致时，以其它文档中的描述为准。

并也适用西门子使用条款 (<https://support.industry.siemens.com>)。

安全提示

西门子为其产品及解决方案提供工业信息安全功能，以支持工厂、系统、机器设备和网络的安全运行。

为了针对网络威胁为工厂、系统、机器设备和网络提供保护，需要实现（并连续保持）一种整体且采用最新技术的工业信息安全方案。西门子的产品与解决方案仅构成这种安全方案的一个元素。

用户的责任是防止人员未经授权而访问其工厂、系统、机器设备和网络。系统、机器设备和组件只应连接到企业网络或互联网，前提是采取必要且适当的安全措施，例如，使用防火墙后进行网络分隔。

如需了解需实施的工业网络与信息安全方面的详细信息，请访问网址

<http://www.siemens.com/industrialsecurity>。

西门子不断对产品和解决方案持续进行开发，以使其更加安全。西门子强烈建议您尽快应用产品更新，并使用最新的产品版本。如果继续使用不再受支持的产品版本，以及没有应用最新更新，可能会增加客户遭受网络威胁的风险。

为了确保及时获知产品更新，请订阅西门子工业信息安全 RSS 源，网址为：

<https://www.siemens.com/industrialsecurity>。

目录

法律声明	2
1 前言	6
2 S7-1200/S7-1500 中的创新	8
2.1 简介	8
2.2 术语	8
2.3 编程语言	11
2.4 优化的机器代码	11
2.5 块创建	12
2.6 优化的块	13
2.6.1 S7-1200: 优化的块的结构	13
2.6.2 S7-1500: 优化的块的结构	14
2.6.3 S7-1500 的处理器优化数据存储	15
2.6.4 优化和未优化的变量之间的转换	18
2.6.5 优化和未优化访问块之间的参数传送	18
2.6.6 使用优化数据进行通信	20
2.7 块属性	21
2.7.1 块的大小	21
2.7.2 组织块 (OB) 的数量	21
2.7.3 块接口 – 隐藏块参数 (V14 或更高版本)	22
2.8 S7-1200/1500 的新数据类型	23
2.8.1 基本数据类型	23
2.8.2 数据类型 Date_Time_Long	24
2.8.3 其它时间数据类型	24
2.8.4 Unicode 数据类型	25
2.8.5 数据类型 VARIANT (S7-1500 和 S7-1200 固件 V4.1 及以上)	26
2.9 指令	29
2.9.1 MOVE 指令	29
2.9.2 VARIANT 指令 (S7-1500 和 S7-1200 FW4.1 及更高版本)	32
2.9.3 RUNTIME	33
2.9.4 PLC 数据类型变量的比较 (V14 或更高版本)	34
2.9.5 多重赋值 (V14 或更高版本)	35
2.10 符号和注释	36
2.10.1 编程编辑器	36
2.10.2 监控表中的注释行	37
2.11 系统常量	38
2.12 用户常量	40
2.13 控制器和 HMI 变量的内部引用 ID	41
2.14 发生错误时的 STOP 模式	43
3 常规编程	44
3.1 操作系统与用户程序	44
3.2 程序块	44
3.2.1 组织块 (OB)	45
3.2.2 函数 (FC)	47
3.2.3 函数块 (FB)	49
3.2.4 实例	50
3.2.5 多重实例	51
3.2.6 以参数形式传送实例 (V14)	53
3.2.7 全局数据块 (DB)	54

3.2.8	无需重新初始化即可下载.....	55
3.2.9	块的可再用性.....	59
3.2.10	块的自动编号.....	60
3.3	块接口类型.....	61
3.3.1	传值.....	61
3.3.2	传引用.....	61
3.3.3	参数传送一览.....	62
3.4	存储器概念.....	62
3.4.1	通过块接口进行数据交换.....	62
3.4.2	全局存储器.....	63
3.4.3	局部存储器.....	64
3.4.4	存储器区域的访问速度.....	65
3.5	保持性.....	66
3.6	符号寻址.....	69
3.6.1	用符号寻址替代绝对寻址.....	69
3.6.2	ARRAY 数据类型和间接域访问.....	71
3.6.3	形式参数 Array[*] (V14 或更高版本).....	73
3.6.4	STRUCT 数据类型和 PLC 数据类型.....	74
3.6.5	通过 PLC 数据类型访问 I/O 区域.....	77
3.6.6	分片访问.....	79
3.6.7	LAD 和 FBD 中的 SCL 程序段 (V14 及更高版本).....	80
3.7	库.....	81
3.7.1	库类型和库元素.....	82
3.7.2	类型概念.....	83
3.7.3	CPU 和 HMI 的可类型化对象的差异.....	84
3.7.4	块的版本控制.....	84
3.8	提高了硬件中断性能.....	89
3.9	其它性能建议.....	91
3.10	SCL 编程语言：提示与技巧.....	92
3.10.1	使用调用模板.....	92
3.10.2	什么指令参数是强制性参数？.....	93
3.10.3	整个变量名称的拖放.....	93
3.10.4	使用关键字 REGION (V14 或更高版本) 结构化.....	94
3.10.5	正确使用 FOR、REPEAT 和 WHILE 循环.....	95
3.10.6	有效使用 CASE 指令.....	96
3.10.7	不操作 FOR 循环的循环计数器.....	96
3.10.8	反向 FOR 循环.....	97
3.10.9	方便地创建实例进行调用.....	97
3.10.10	时间变量的处理.....	97
3.10.11	不必要的 IF 指令.....	99
4	编程不依赖于硬件.....	100
4.1	S7-300/400 和 S7-1200/1500 的数据类型.....	100
4.2	不使用位存储器而使用全局数据块.....	102
4.3	“时钟位”编程.....	102
5	TIA Portal 中的 STEP 7.....	103
5.1	简介.....	103
5.2	术语.....	104
5.3	安全程序的组件.....	105
5.4	F 运行组.....	106
5.5	F 签名.....	106
5.6	分配 F-I/O 的 PROFIsafe 地址.....	108

5.7	F-I/O 的检测.....	108
5.8	值状态 (S7-1200F/1500F).....	109
5.9	数据类型.....	110
5.9.1	概述.....	110
5.9.2	隐式转换.....	110
5.10	符合 F 的 PLC 数据类型.....	112
5.11	TRUE / FALSE.....	114
5.12	优化编译和程序运行.....	115
5.12.1	避免时间处理块: TP, TON, TOF.....	116
5.12.2	避免深度调用层次.....	116
5.12.3	避免 JMP/LABEL 结构.....	116
5.13	标准程序和 F 程序之间的数据交换.....	117
5.14	测试安全程序.....	118
5.15	发生 F 错误时的 STOP 模式.....	119
5.16	移植安全程序.....	119
5.17	一般安全建议.....	119
6	使用用户程序自动生成可视化.....	120
6.1	简介.....	120
6.2	如何实现自动生成.....	121
6.3	控制 HMI 生成器.....	122
6.3.1	使用程序段注释进行控制.....	122
6.3.2	使用 SiVArc 变量进行控制.....	123
6.4	其它建议.....	124
7	最重要的建议.....	125
8	附录.....	126
8.1	服务与支持.....	126
8.2	链接与文献.....	127
8.3	文档变更.....	128

1 前言

新一代 SIMATIC 控制器的开发目标

- 所有自动化组件（控制器、HMI、传动等）的工程平台
- 统一编程
- 提高性能
- 兼容各种编程语言指令
- 完全符号编程
- 数据处理也无需指针
- 创建的块可再用

该编程指南的目标

新一代控制器 SIMATIC S7-1200 和 S7-1500 具有最新的系统架构，可与 TIA Portal 一起实现全新高效编程和组态。这样，无需关注控制器资源（例如存储器中的数据存储），重点聚焦在实际的自动化解方案本身。

本文为您提供了许多关于 S7-1200/1500 控制器最佳编程的建议和注意事项。本文对 S7-300/400 系统架构中的一些差异，以及由此相关的新编程选项以易于理解的方式进行了解释。这样有助于您为自动化解方案创建标准化的最佳编程。

所述的示例可以通用于控制器 S7-1200 和 S7-1500。

本编程指南的核心内容

本文讨论了以下有关 TIA Portal 的关键问题：

- S7-1200/1500 中的创新
 - 编程语言
 - 优化的块
 - 数据类型与指令
- 通用编程建议
 - 操作系统与用户程序
 - 存储器概念
 - 符号寻址
 - 库
- 有关独立于硬件的编程建议
- 关于 TIA Portal 中 STEP 7 Safety 的建议
- 最重要建议一览

优点和益处

遵循这些建议和技巧会带来许多益处：

- 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 控制器编程时，编程人员的任务是尽可能清晰易读地编制用户程序。每个用户都使用他们自己的策略，例如，如何命名变量、块或注释的方式。编程人员的不同编程理念创造了非常不同的用户程序，但只能由各自的编程人员来解读。

编程风格指南提供了一套协调一致的编程规则。例如，这些规范描述了变量和块名的统一赋值，直至在 SCL 中进行清晰的编程。

您可以自由应用这些规则和建议；将其作为一致编程的建议（而非编程的标准）。

注

S7-1200 和 S7-1500 编程风格指南参见以下链接：
<https://support.industry.siemens.com/cs/ww/en/view/81318674>

2 S7-1200/S7-1500 中的创新

2.1 简介

一般而言，从 S7-300/400 到 S7-1500，SIMATIC 控制器的编程都保持了不变。有已知的编程语言，例如 LAD、FBD、STL、SCL 或 Graph 和块，例如组织块 (OB)、函数块 (FB)、函数 (FC) 或数据块 (DB)。S7-300/400 程序可以很容易地在 S7-1500 上实现，现有 LAD、FBD 和 SCL 程序也都可很容易地在 S7-1200 控制器上实现。

此外，还有许多创新可以帮助您进行编程，并支持强大且节省存储器的代码。

对于为 S7-1200/1500 控制器实现的程序，我们建议不要一对一地编程，而是检查是否有新的选项，如有可能，使用这些选项。通常不需要额外的付出，即可实现符合下列情况的程序代码：

- 对新 CPU 的存储器和运行时来说最佳
- 更易理解
- 更易维护

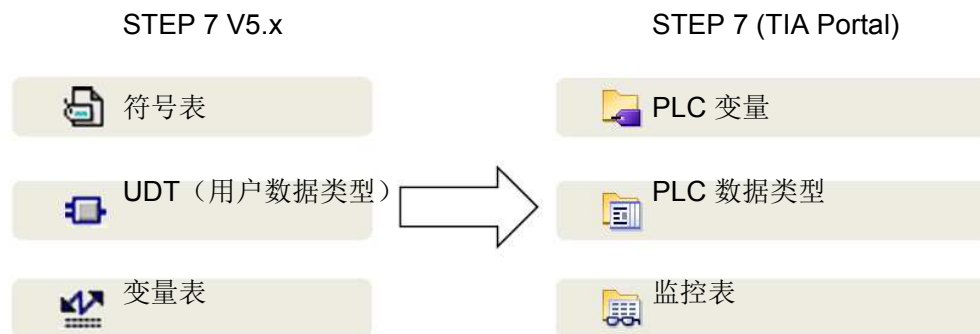
注 S7-300/S7-400 到 S7-1500 的迁移信息参见以下条目：
<https://support.industry.siemens.com/cs/ww/en/view/109478811>

2.2 术语

TIA Portal 中的常用术语

一些术语已变更，使 TIA Portal 更容易处理。

图 2-1: TIA Portal 中的新术语



变量和参数术语

在处理变量、函数和函数块时，许多术语被重复地以不同的方式使用，甚至被错误地使用。下图阐明了这些术语。

图 2-2：变量和参数术语

全局数据块

MainOBGlobal			
	Name	Data type	Start value
1	Static		
	statMoveVariantEnable	nt	0
	statInputInt	nt	15
	statOutputInt	nt	0
	statInputReal	Real	17.3
	statOutputReal	Real	0.0
	statInputMyType	'MyType'	
	statOutputMyType	'MyType'	
	statInputBool	bool	false
	statError	nt	0

函数/函数块

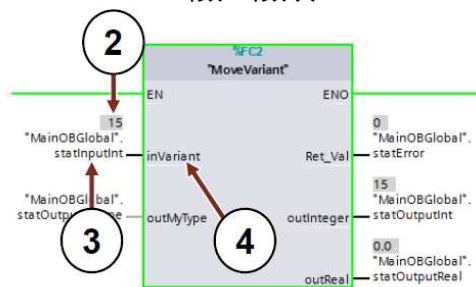


表 2-1：变量和参数术语

	术语	说明
1.	变量	变量由名称/标识符标记，并使用控制器存储器中的地址。变量始终用特定的数据类型（布尔型、整数型等）来定义： <ul style="list-style-type: none"> • PLC 变量 • 数据块中的单个变量 • 整个数据块
2.	变量值	变量值是存储在变量中的值（例如，15 表示整数变量的值）。
3.	实际参数	实际参数是在指令、函数和函数块的接口处相互连接的变量。
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

更多信息，参见以下条目：

网络上有哪些内容信息可用于迁移到 STEP 7 (TIA Portal) 和 WinCC (TIA Portal) ?

<https://support.industry.siemens.com/cs/ww/en/view/56314851>

在 STEP 7 专业版 (TIA Portal) 中迁移 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 Portal) 将 PLC 迁移到 S7-1500:

<https://support.industry.siemens.com/cs/ww/en/view/67858106>

您如何在 S7- 1200/S7-1500 STEP 7 (TIA Portal) 中高效且有效地编程？

<https://support.industry.siemens.com/cs/ww/en/view/67582299>

为什么无法将寄存器传递和显式参数传递与 STEP 7 (TIA Portal) 中的 S7-1500 混用？

在其它主题中，本条目描述了 STL 程序到 S7-1500 的迁移。

<https://support.industry.siemens.com/cs/ww/en/view/67655405>

2.3 编程语言

用户程序可使用不同的编程语言进行编程。每种语言都有自己的优点，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 2-2: 编程语言

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

注

更多信息，参见以下条目：

基于国际助记法的 SIMATIC S7-1200/S7-1500 编程语言对照表

<https://support.industry.siemens.com/cs/ww/en/view/86630375>

在 STEP 7 (TIA Portal) 中迁移 S7-SCL 程序时，您应该注意什么？

<https://support.industry.siemens.com/cs/ww/en/view/59784005>

在 STEP 7 (TIA Portal) 的 SCL 程序中，您无法使用哪些指令？

<https://support.industry.siemens.com/cs/ww/en/view/58002709>

在 STEP 7 (TIA Portal) 中，您如何定义 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 Portal 和 S7-1200/1500 支持每种编程语言的优化运行时性能。所有语言都是以同样的方式直接用机器代码编译的。

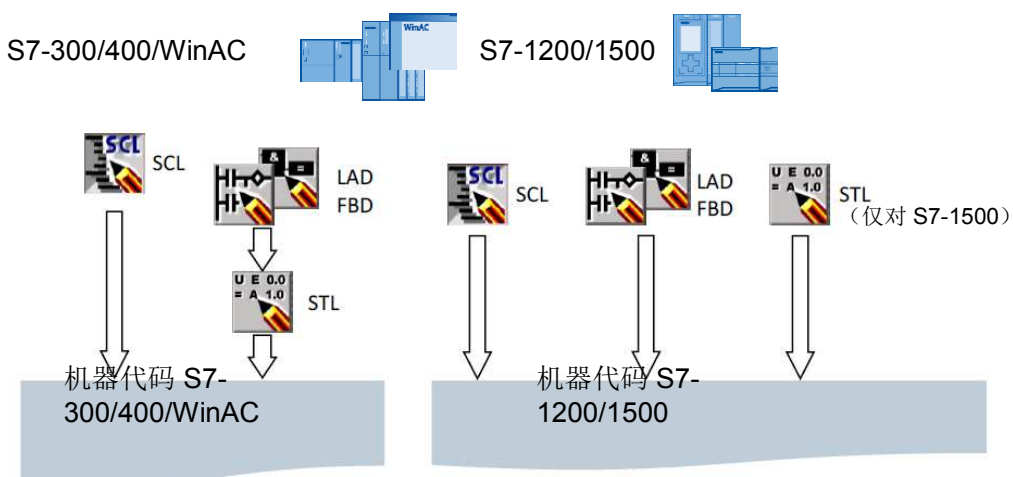
优点

- 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而言）
- 通过 STL 的中间步骤的额外编译不会降低性能

属性

下图显示了机器代码 S7 程序编译的差异。

图 2-3: 使用 S7-300/400/WinAC 和 S7-1200/1500 创建机器代码

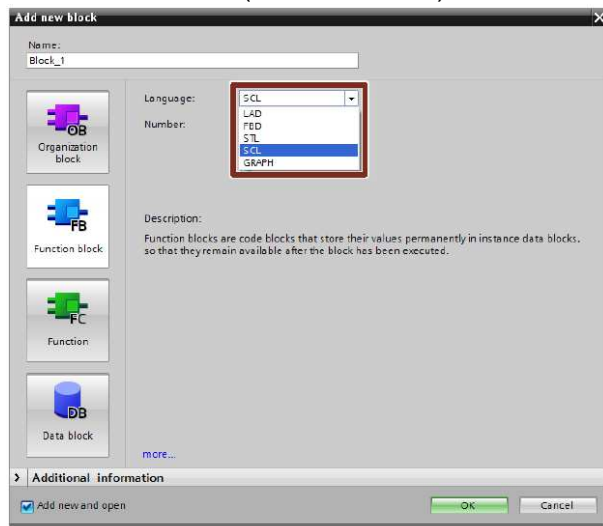


- 对于 S7-300/400/WinAC 控制器，在创建机器代码之前，首先在 STL 中编译 LAD 和 FBD 程序。
- 对于 S7-1200/1500 控制器，所有编程语言都直接用机器代码编译。

2.5 块创建

所有块，如 OB、FB 和 FC，都可以用所需的编程语言直接编程。因此，不需要为 SCL 程序创建任何源。仅选择块和 SCL 作为编程语言。然后，可以直接对块进行编程。

图 2-4: “添加新块”(Add new Block) 对话框



2.6 优化的块

S7-1200/1500 控制器具有优化的数据存储。在优化的块中，所有变量都根据其数据类型自动排序。排序确保了变量之间的数据间隙减至最小，并且确保变量针对处理器进行访问优化存储。

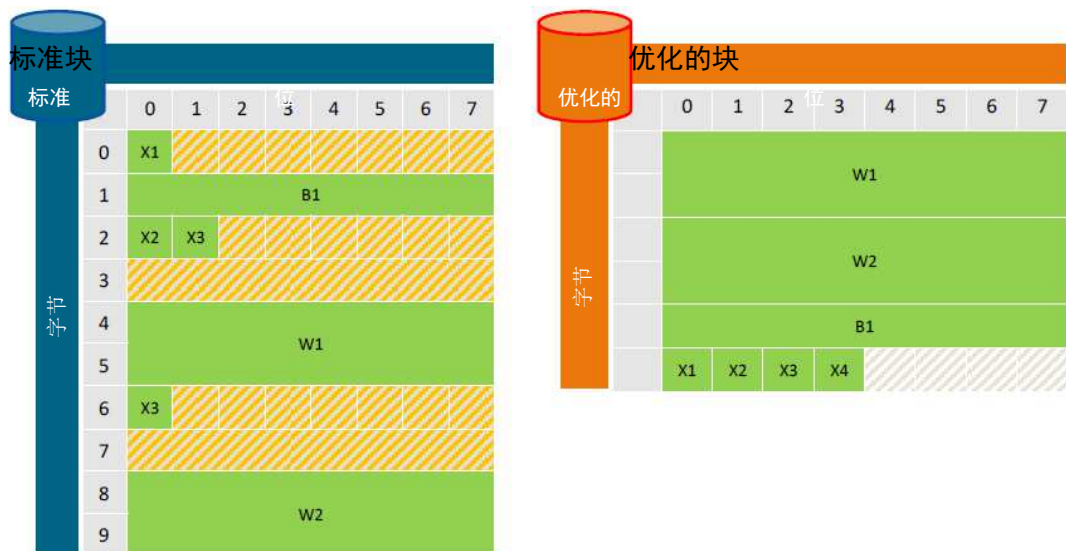
未优化的块仅在 S7-1200/1500 控制器中出于兼容性原因时可用。

优点

- 始终尽可能快地进行访问，因为数据存储是由系统优化的，并且与声明无关。
- 由于访问通常是符号访问，因此，没有由于错误的绝对访问而导致不一致的危险。
- 声明变更不会导致访问错误，因为，例如，HMI 访问是符号访问。
- 单独变量可以明确定义为保持变量。
- 背景数据块中不需要设置。各项都设置在指定的函数块中（例如，保持项目）。
- 数据块中的存储预留允许在不损失当前值的情况下进行更改（参见第 3.2.8 节“无需重新初始化即可下载”）。

2.6.1 S7-1200：优化的块的结构

图 2-5：优化的块 (S7-1200)



属性

- 未形成数据间隙，因为较大的变量位于块的开头，较小的变量位于块的结尾。
- 优化的块只支持符号访问。

2.6.2 S7-1500: 优化的块的结构

图 2-6: 优化的块 (S7-1500)

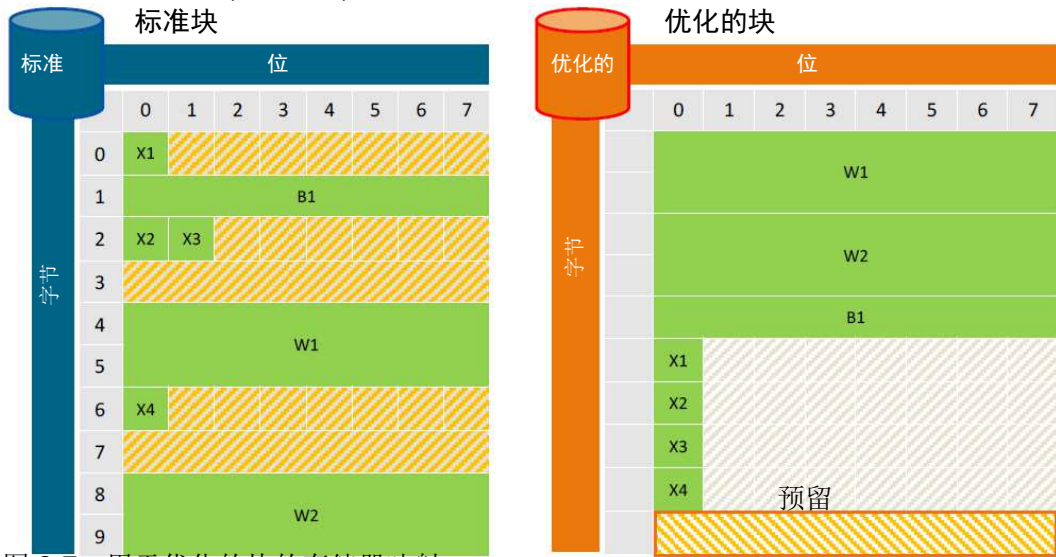
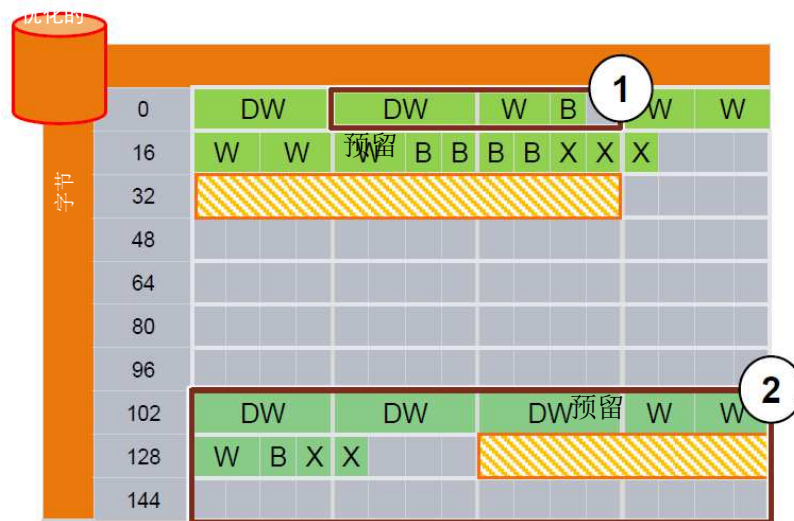


图 2-7: 用于优化的块的存储器映射



1. 各结构单独设置，因此，可以作为块复制。
2. 保持数据位于单独的区域，可以作为块复制。掉电时，这些数据将保存在 CPU 内部。“MRES”将该数据重置为装载存储器中的起始值。

属性

- 未形成数据间隙，因为较大的变量位于块的开头，较小的变量位于块的结尾。
- 由于处理器优化存储，访问速度更快（所有变量合理存储，使得 S7-1500 的处理器只需一个机器指令即可直接读取或写入变量）。
- 布尔型变量存储为字节，以便更快地访问。因此，控制器不必屏蔽访问。

- 优化的块有一个存储预留，用于在运行操作中进行装载（参见第 3.2.8 节“无需重新初始化即可下载”）。
- 优化的块只支持符号访问。

2.6.3 S7-1500 的处理器优化数据存储

出于与第一款 SIMATIC 控制器兼容的原因，S7-300/400 控制器接受“大端格式”数据存储的原则。

基于改变后的处理器架构，新一代 S7-1500 控制器始终以“小端格式”序列访问 4 个字节（32 位）。因此，系统端产生以下属性。

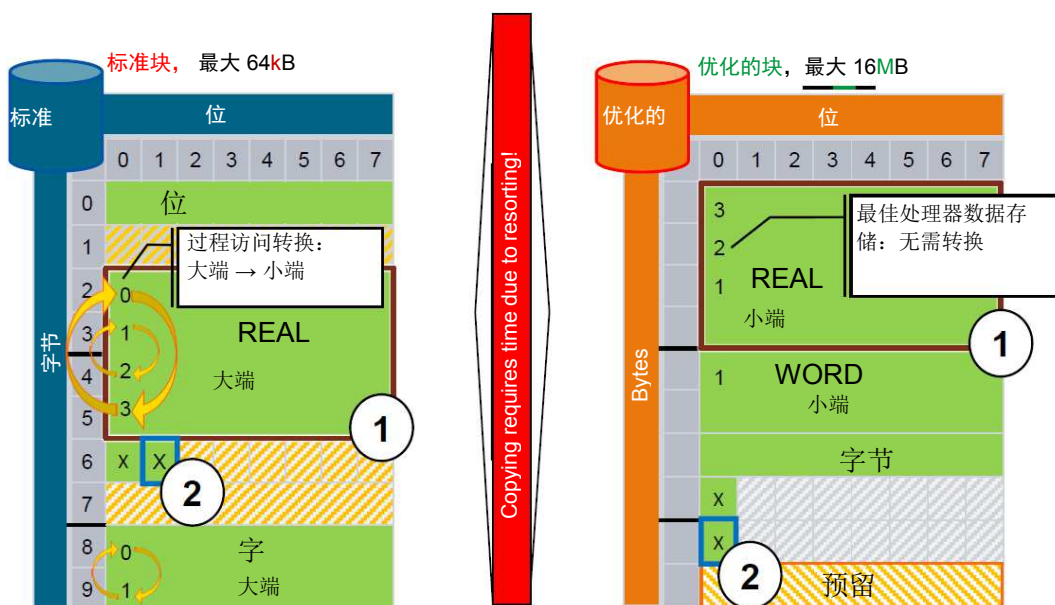


表 2-3: S7-1500 控制器的数据访问

	标准块	优化的块
1.	在出现不利偏移时，控制器需要 2x16 位访问，来读取 4 字节值（例如，REAL 值）。此外，必须转换字节。	控制器存储访问优化的变量。32 位 (REAL) 访问。无需转换字节。
2.	每个位访问读取并屏蔽整个字节。进行任何其它访问时，完整字节锁闭。	每个位均分配一个字节。控制器在访问时不必屏蔽字节。
3.	最大块大小为 64kB。	最大块大小为 16MB。

建议

- 通常，只使用优化的块。
 - 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。符号数据也可以间接寻址（参见第 3.6.2 节 ARRAY 数据类型和间接字段访问）。
 - 在控制器中处理优化的块比处理标准块快得多。
- 避免在优化和未优化的块之间复制/赋值数据。源格式和目标格式之间所需的数据转换需要很长的处理时间。


示例：设置优化的块访问

默认情况下，对 S7-1200/1500 的所有新创建的块启用优化的块访问。可以为 OB、FB 和全局 DB 设置块访问。例如，DB，从相应的 FB 派生设置。

如果块从 S7-300/400 控制器迁移到 S7-1200/1500，则块访问不会自动重置。可以稍后将块访问更改为“优化的块访问”。更改块访问后，必须重新编译程序。如果将 FB 更改为“优化的块访问”，则赋值的背景数据块将自动更新。

按照指令设置优化的块访问。

表 2-4：设置优化的块访问

步骤	说明
1.	单击项目树中的“最大化/最小化概览图”(Maximizes/minimizes the Overview) 按钮。 
2.	导航至“程序块”(Program blocks)。

说明	
3.	<p>在此，可以查看程序中的所有块，以及它们是否经过优化。在本概览图中，可以方便地更改“优化的块访问”(Optimized block access) 状态。</p>  <p>注：背景数据块（此处为“Function_block_1_DB”）从相关的 FB 承继“优化”状态。这就是为什么“优化”设置只能在 FB 上更改的原因。在项目编译之后，DB 根据相关的 FB 呈现状态。</p>

TIA Portal 中优化和未优化的块的显示

在下图中，可以看到优化背景数据块和未优化背景数据块之间的差异。对于全局数据块而言，差异相同。

图 2-9: 优化的数据块（无偏移）

InstLGF_PulseRelay			
	Name	Data type	Start value
1	Input		
2	trigger	Bool	false
3	set	Bool	false
4	reset	Bool	false
5	Output		
6	out	Bool	false

图 2-10: 未优化的数据块（有偏移）

InstLGF_PulseRelay				
	Name	Data type	Offset	Start value
1	Input			
2	trigger	Bool	0.0	false
3	set	Bool	0.1	false
4	reset	Bool	0.2	false
5	Output			
6	out	Bool	2.0	false

表 2-5: 差别：优化的数据块和未优化的数据块

优化的数据块	未优化的数据块
--------	---------

优化的数据块通过符号访问寻址。因此，没有显示“偏移”(offset)。	对于未优化的块，显示“偏移”(offset)，并可用于寻址。
在优化的块中，可以用“预留”(Retain)来单独声明每个变量。	在未优化的块中，用“预留”(Retain)只可声明全部变量或不得声明变量两种形式。

全局数据块的变量保持性直接在全局数据块中定义。默认情况下，非保持性是预设的。定义函数块实例（而非背景数据块）中变量的保持性。因此，这些设置对该 FB 的所有实例都有效。

优化和未优化数据块的访问类型

下表显示了块的所有访问类型。

表 2-6: 访问类型

访问类型	优化的块	未优化的块
符号	√	√
变址（字段）	√	√
分片访问	√	√
AT 指令	-（可选：分片访问）	√
直接绝对访问	-（可选：ARRAY, INDEX）	√
间接绝对访问（指针）	-（可选：VARIANT /ARRAY, 带变址）	√
无需重新初始化即可装载	√	-

注

更多信息，参见以下条目：

在 STEP 7 (TIA Portal) 中，哪些类型的访问可用于访问块中的数据值，并且应该注意哪些类型之间的差异？

<https://support.industry.siemens.com/cs/ww/en/view/67655611>

在STEP 7 (TIA Portal) 中，当使用具有优化访问功能的数据库时，对于“READ_DBL”和“WRIT_DBL”指令，应该注意哪些属性？

<https://support.industry.siemens.com/cs/ww/en/view/51434747>

2.6.4 优化和未优化的变量之间的转换

通常建议使用优化的变量。然而，如果在个别情况下，想保持您的编程到目前为止，将优化和未优化的混合数据存储存储在程序中。

系统了解每个变量的内部存储状态，不论是结构化的（从单独定义的数据类型得到）还是基本的（INT, LREAL, ...）。

对于不同存储器存储的两个变量之间的相同类型的分配，系统会自动转换。这种转换需要结构化变量的性能，因此应该尽可能避免。

2.6.5 优化和未优化访问块之间的参数传送

当将结构作为输入/输出参数 (InOut) 传送到被调用的块时，它们在默认情况下是作为引用参数传送的（参见第 3.3.2 节“传引用”）。

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，情况则不同。在这种情况下，所有参数通常都作为副本传送（参见第 3.3.1 节“传值”）。

此时，被调用的块始终使用复制的值。在块处理期间，在块调用处理之后，这些值可能会被更改，并被复制回原始操作数。

如果原始操作数被异步进程（例如，HMI 访问或中断 OB）更改，则可能会导致问题。如果在块处理之后副本被复制回原始操作数，则对原始操作数异步执行的更改将被覆盖。

注

更多信息，参见以下条目：

为什么在 S7-1500 中有时会覆盖 HMI 系统或 web 服务器的数据？

<https://support.industry.siemens.com/cs/ww/en/view/109478253>

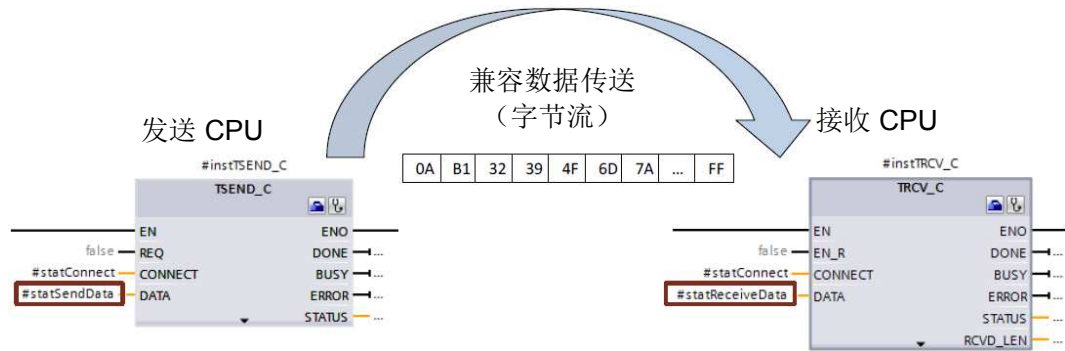
建议

- 应始终为相互通信的两个块设置相同的访问类型。

2.6.6 使用优化数据进行通信

接口（CPU、CM）以其既定的方式传送数据（不管是否被优化）。

图 2-11: CPU-CPU 间通信



发送数据可以是：

- 优化的
- 未优化的
- 变量（任何类型）
- 缓冲区（字节数组）

接收数据可以是：

- 优化的
- 未优化的
- 变量（任何类型）
- 缓冲区（字节数组）

示例

- PLC 数据类型（数据记录）的变量应传递给 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，接收数据被分配给相同类型的变量。
- 此时，接收数据的符号可以直接继续适用。

注 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 还可以以不同方式定义发送和接收数据：

发送数据 接收数据

优化的 --> 未优化的

未优化的 --> 优化的

控制器自动确保正确传送和存储数据。

2.7 块属性

2.7.1 块的大小

对于 S7-1200/1500 控制器，主存储器中的最大块大小明显增大。

表 2-7: 块的大小

最大容量和数量（不考虑存储器大小）		S7-300/400	S7-1200	S7-1500
DB	最大大小	64 kB	64 kB	64 kB 16 MB（优化的 CPU1518）
	最大数量	16 000	65 535	65 535
FC / FB	最大大小	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)

建议

- 将 S7-1500 控制器的数据库用作大量数据的数据容器。
- 使用 S7-1500 控制器，可以将大于 64 kB 的数据量存储在优化的 DB 中（最大容量为 16 MB）。

2.7.2 组织块 (OB) 的数量

使用 OB 可以创建用户程序的层次结构。为此提供有各种不同的 OB。

表 2-8: 组织块的数量

组织块类型	S7-1200	S7-1500	优点
循环和启动 OB	100	100	用户程序的模块化
硬件中断	50	50	每个事件都可能有一个单独的 OB
延迟中断	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	-	20	用户程序的模块化

*从固件 V4 开始，可能有 4 个延迟中断和 4 个循环中断。

建议

- 使用 OB 来结构化用户程序。
- 关于 OB 使用的进一步建议，参见第 3.2.1 节“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，可以具体显示或隐藏块参数。在此，可以为每个形式参数单独设置三个选项。

- “显示”(Show)
- “隐藏”(Hide)
- “未指定参数时隐藏”(Hide if no parameter is assigned)

优点

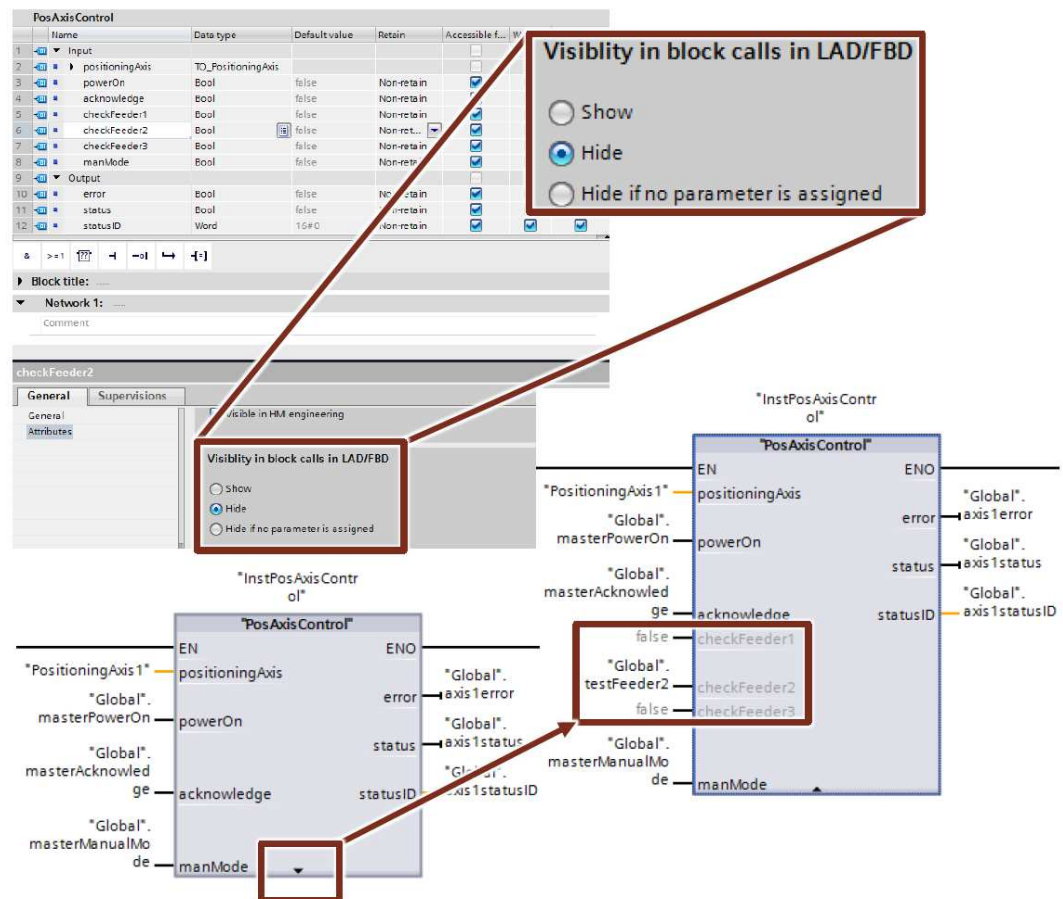
- 更佳地概览具有许多可选参数的块

属性

- 可应用于：
 - FC, FB
 - In, Out, InOut

示例

图 2-12: 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 更多信息，参见以下条目：
在 STEP 7 (TIA Portal) ，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 2-9: 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
ULInt*	64 位	0 .. 18.4 Trio (1018)
LInt*	64 位	-9.2 Trio .. 9.2 Trio
LWord	64 位	16#0000 0000 0000 0000 ~ 16# FFFF FFFF FFFF FFFF

*仅适用于 S7-1500

表 2-10: 浮点型数据类型

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

注 更多信息，参见以下条目：
在 STEP 7 (TIA Portal) 中，为什么 SCL 的 DInt Addition 结果没有正确显示？
<https://support.industry.siemens.com/cs/ww/en/view/98278626>

2.8.2 数据类型 Date_Time_Long

表 2-11: DTL 的结构 (Date_Time_Long)

年	月	日	工作日	小时	分钟	秒	毫微秒
---	---	---	-----	----	----	---	-----

DTL 始终读取当前系统时间。通过符号名称（例如，My_Timestamp.Hour）对具体值进行访问

优点

- 所有的子区域（例如年、月、...）都可以通过符号访问处理。

建议

使用新的数据类型 DTL 替代 LDT，并通过符号访问对其进行寻址（例如，My_Timestamp.Hour）。

注

更多信息，参见以下条目：

在 STEP 7 (TIA Portal)，如何输入、读取和编辑 S7-300/S7-400/S7-1200/S7-1500 CPU 块的日期和时间？

<https://support.industry.siemens.com/cs/ww/en/view/43566349>

在 STEP 7 V5.5 和 TIA Portal 中，哪些功能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 其它时间数据类型

表 2-12: 时间数据类型（仅针对 S7-1500）

类型	大小	数值范围
LTime	64 位	LT#- 106751d23h47m16s854ms775us808ns ~ LT#+106751d23h47m16s854ms775us8 07ns
LTIME_OF_DAY	64 位	LTOD#00:00:00.000000000 ~ LTOD#23:59:59.999999999

2.8.4 Unicode 数据类型

借助于数据类型，可以处理 WCHAR 和 WSTRING Unicode 字符。

表 2-13: 时间数据类型（仅针对 S7-1500）

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

属性

- 例如，对拉丁语、汉语或其它语言中的字符的处理。
- 换行符、换页符、制表符、空格
- 特殊字符：美元符号，引号

示例

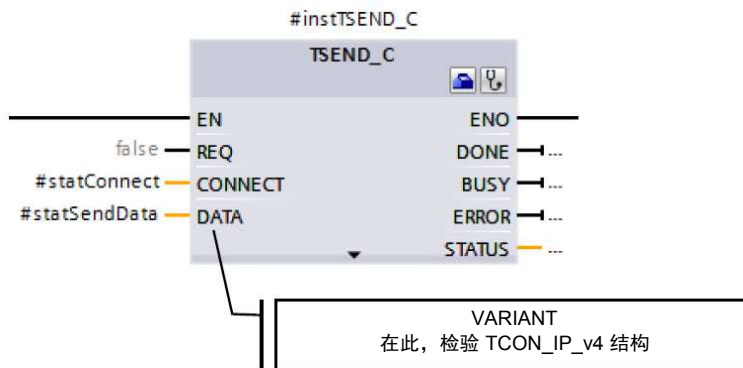
- WCHAR#'a'
- WSTRING#'Hello World!'

2.8.5 数据类型 VARIANT (S7-1500 和 S7-1200 固件 V4.1 及以上)

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 指针不同，VARIANT 是具有型式检验的指针。这意味着，目标结构和源结构在运行时被检验，并且必须相同。

例如，VARIANT 用作通信块 (TSEND_C) 的输入。

图 2-13: 用作指令 TSEND_C 输入参数的数据类型 VARIANT



优点

- 集成型式检验可防止错误访问。
- 通过不同变量的符号寻址，可以更容易地读取代码。
- 代码效率更高，时间更短。
- Variant 指针显然比 ANY 指针都更直观。
- 借助系统功能，可以直接使用正确类型的 variant 变量。
- 可实现不同结构化变量的灵活、高效传送。

属性

比较 ANY 和 Variant 时，可以查看属性。

表 2-14: 比较 ANY 和 Variant

ANY	Variant
要求 10 字节的存储器，具有既定的结构	无需用户的主存储器
通过赋值数据区或填充 ANY 结构进行初始化	通过赋值数据区或系统指令进行初始化
非类型化 – 无法识别互连结构的类型	类型化 – 互连类型，对于数组，也可以确定长度
部分类型化 – 对于数组，也可以确定长度	VARIANT 可以通过系统指令进行检测和创建

建议

- 在必须使用 **ANY** 指针之前，请检查位置。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当时确定时，数据类型 **VARIANT** 仅用于间接寻址。
 - 使用数据类型 **VARIANT** 作为 **InOut** 形式参数来创建独立于实际参数的数据类型的通用块（参见本章中的示例）。
 - 使用 **VARIANT** 数据类型，而非 **ANY** 指针。借助集成型式检验，在早期即可检测到错误。由于符号寻址，程序代码很容易解读。
 - 例如，使用 **variant** 指令进行类型识别（参见以下示例和第 2.9.2 节 **VARIANT** 指令）
- 使用数组的索引，而非通过 **ANY** 来寻址数组元素（参见第 3.6.2 节 **ARRAY** 数据类型和间接字段访问）。

表 2-15: 比较 **ANY** 指针和简化形式

ANY 指针的用途是什么？		S7-1200/1500 简化形式
可以处理不同数据类型的编程函数	→	将 variant 指针用作块 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	→	默认数组函数 • 读取和写入 #myArray[#index] (参见第 3.6.2 节 ARRAY 数据类型和间接字段访问) • 使用 MOVE_BLK 复制 (参见第 2.9.1 节 MOVE 指令)
• 通过绝对寻址传送结构和处理性能 例如，通过函数 ANY 指针来传递用户定义的结构	→	将结构转换为 InOut 参数 • 参见第 3.3.2 节“传引用”

注

如果要复制非结构化 **VARIANT** 变量的值，还可以使用 **VariantGet** 代替 **MOVE_BLK_VARIANT** (第 2.9.2 节 **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，并相应地做出响应。函数“**MoveVariant**”的以下代码显示了一段可行的编程。

- **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。
- 实际参数的数据类型用“**Type_Of**”指令检测
- 根据数据类型的不同，变量值会通过“**MOVE_BLK_VARIANT**”指令复制到不同的输出形式参数中。
- 如果未检测到实际参数的数据类型，则该块将输出一个错误代码。

图 2-14: 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: // 移动整数
    #MoveVariant := MOVE_BLK_VARIANT(SRC := #inOutVariant,
      COUNT := 1,
      SRC_INDEX := 0,
      DEST_INDEX := 0,
      DEST => #outInteger);

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

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

ELSE // 出错，没有足够的数据类型
  #MoveVariant := WORD_TO_INT(#NO_CORRECT_DATA_TYPE);
  // 80B4: MOVE_BLK_VARIANT 的错误代码：数据类型不对应
END_CASE;
```

2.9 指令

TIA Portal 为程序员提供了现成的指令（位逻辑、时间、计数器、比较器.....）。

注

更多函数可在以下条目中下载：

STEP 7 (TIA Portal) 和S7-1200/S7-1500 (LGFP) 的通用函数库

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

2.9.1 MOVE 指令

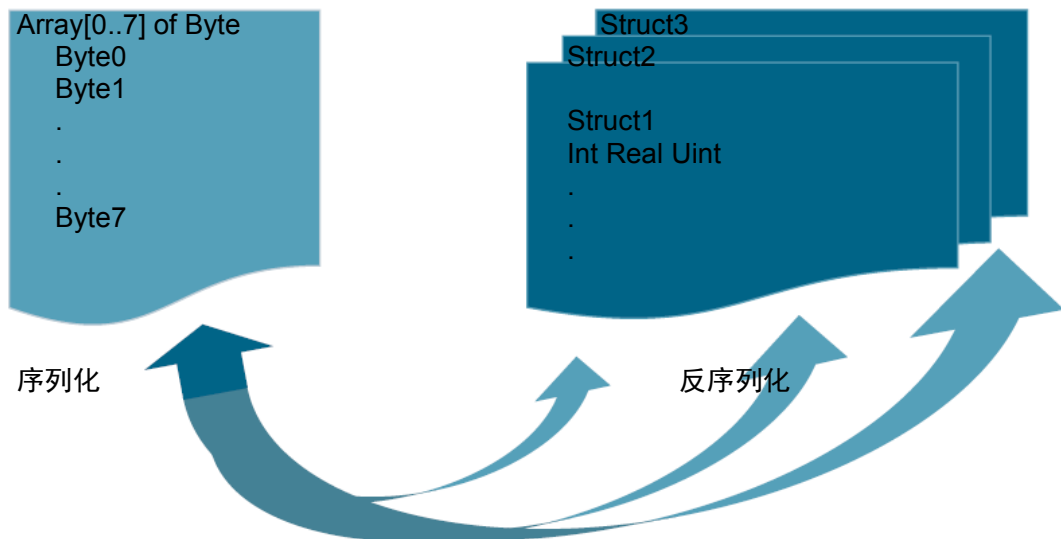
在 STEP 7 (TIA Portal) 中，以下 MOVE 指令可用。MOVE_BLK_VARIANT 指令是 S7-1200/1500 的新增指令。

表 2-16: MOVE 指令

指令	用途	属性
MOVE	复制值	<ul style="list-style-type: none">将输入 IN 的参数内容复制到输出 OUT 的参数中。输入和输出的参数必须是相同的数据类型。参数也可以是结构化变量（PLC 数据类型）。复制整个数组和结构。
MOVE_BLK	复制数组	<ul style="list-style-type: none">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。复制整个数组和结构。还可复制若干结构的数组元素。此外，可以指定元素的开始位置和数量。
UMOVE_BLK	不间断地复制数组	<ul style="list-style-type: none">一致地复制数组的内容，而不会有操作中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	<ul style="list-style-type: none">复制一个或多个结构化变量（PLC 数据类型）运行时识别数据类型提供详细的错误信息除了基本和结构化数据类型，还支持 PLC 数据类型、数组和数组数据库。

指令	用途	属性
序列化(Serialize) (S7-1500 和 S7-1200 FW4.1 或更高版本)	将结构化数据转换成字节数组	<ul style="list-style-type: none"> 若干数据记录可以组合成一个字节数组，例如，作为一个消息帧发送到其它设备。 输入和输出参数可以作为数据类型 Variant 进行传送。
反序列化(Deserialize) (S7-1500 和 S7-1200 FW4.1 或更高版本)	将一个字节数组转换成一个或多个结构	<ul style="list-style-type: none"> 应用示例 I-Device: I 设备接收输入区域中的若干数据记录，并将这些记录复制到不同的结构中。 若干数据记录可以组合成一个字节数组。反序列化允许将这些复制到不同的结构。

图 2-15: 序列化和反序列化 (S7-1500 和 S7-1200 FW4.1 或更高版本)



属性

“序列化”、“反序列化”、“CMP”（比较器）和“MOVE：复制值”等指令可以处理非常大和复杂的结构化变量。在这个过程中，CPU 在运行时分析变量结构。处理时间取决于待处理变量结构的以下属性：

- 结构的复杂性
- 不使用 PLC 数据类型的结构数量
- 字节数组可以保存在优化的块中（V14 或更高版本）。

建议

- 使用 PLC 数据类型而不是“STRUCT”来声明结构
- 减少使用的结构数量：
 - 例如，避免重复声明组成非常相似的结构。将这些结构结成一个单一的结构。
 - 当结构的许多元素具有相同的数据类型时，如有可能，使用 ARRAY 数据类型。
- 一般来说，需要区分 MOVE、MOVE_BLK 和 MOVE_BLK_VARIANT
 - 使用 MOVE 指令复制完整的结构。

-
- 使用 `MOVE_BLK` 指令复制已知数据类型的 `ARRAY` 部分。
 - 如果希望复制数组中只有在程序运行时才知悉的数据类型的 `ARRAY` 部分，请只使用 `MOVE_BLK_VARIANT` 指令。

注 **UMOVE_BLK:** 此复制操作不能被其它操作系统任务打断。因此，在指令“无中断复制数组”的处理过程中，CPU 的报警反应时间可能会增加。
有关 `MOVE` 指令的完整描述，请参考 `TIA Portal` 在线帮助。

注 更多信息，参见以下条目：
如何在 `STEP 7 (TIA Portal)` 中复制存储器区域和结构化数据？
<https://support.industry.siemens.com/cs/ww/en/view/42603881>

2.9.2 VARIANT 指令（S7-1500 和 S7-1200 FW4.1 及更高版本）

表 2-17: MOVE 指令

指令	用途	属性
MOVE 指令		
VariantGet	读取数值	该指令使您能够读取指向 VARIANT 的变量值。
VariantPut	写入值	该指令使您能够编写指向 VARIANT 的变量值。
枚举		
CountOfElements	计数元素	使用该指令，可以轮询指向 VARIANT 的变量的 ARRAY 元素数量。
比较器指令		
TypeOf() (仅 SCL)	确定数据类型	使用该指令，可以轮询指向 VARIANT 的变量的数据类型。
TypeOfElements() (仅 SCL)	确定数组数据类型	使用该指令，可以轮询指向 VARIANT 的变量的 ARRAY 元素数据类型。

指令	用途	属性
比较器指令		
VARIANT_TO_DB_A NY (仅 SCL)	确定数据块编号	该指令可以查询 PLC 数据类型、系统数据类型或数组数据库的实例数据块的数据块号。
DB_ANY_TO_VARIAN T (仅 SCL)	从 variant 变量创建数据块。	该指令可以创建 PLC 数据类型、系统数据类型或数组数据库的实例数据块的 variant 变量。

注 有关更多 VARIANT 指令的信息，请参考 TIA Portal 在线帮助。

属性

由于算法复杂，variant 指令比直接指令需要更长的处理时间。

建议

- 如有可能，请不要在循环中使用不同的指令（FOR，WHILE...），以防止增加不必要的循环时间。
- 请不要通过元素的循环来复制数组，而是直接赋值整个数组。

2.9.3 RUNTIME

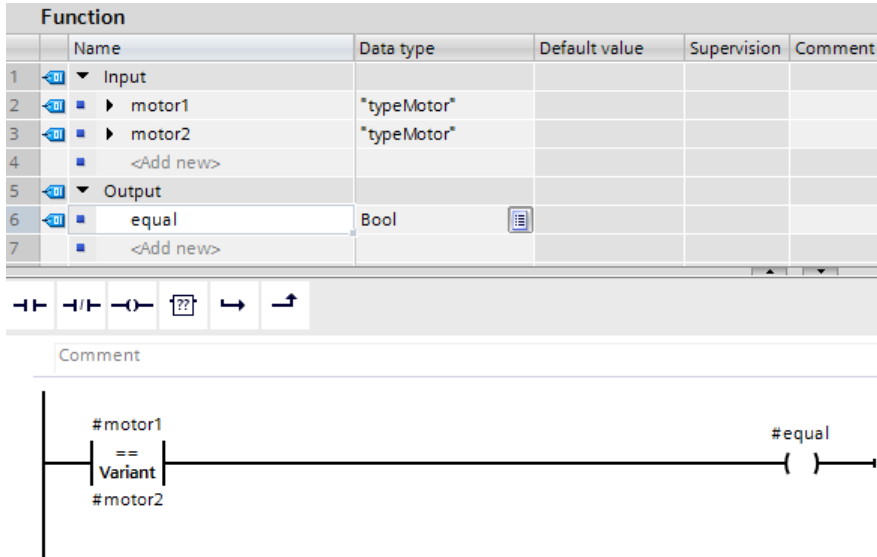
“RUNTIME”指令可以衡量整个程序、单个块或指令序列的运行时间。可在 LAD、FBD、SCL 和 STL（仅 S7-1500）中调用该指令。

注 更多信息，参见以下条目：
使用 S7-1200/S7-1500，如何衡量组织块的总循环时间？
<https://support.industry.siemens.com/cs/ww/en/view/87668055>

2.9.4 PLC 数据类型变量的比较 (V14 或更高版本)

可以检查相同 PLC 数据类型的两个变量的相似性或不同性。

图 2-16: 梯形图中 PLC 数据类型变量的比较

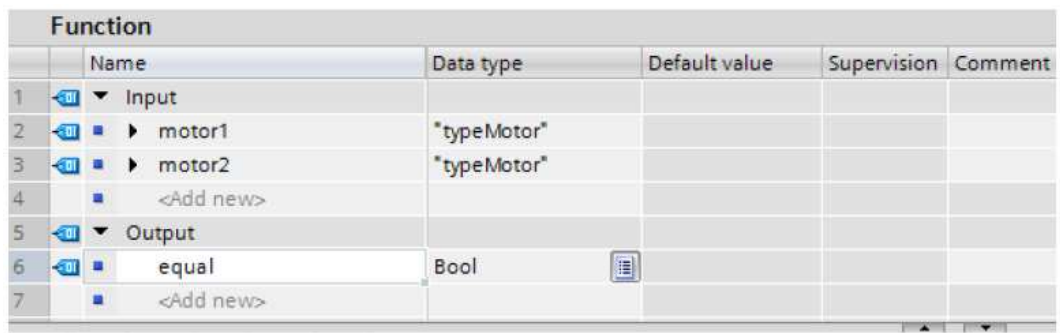


优点

- 结构化变量的符号编程
- 最佳性能的比较
- 可在 LAB、FBD、STL 中进行比较。
- 在 STL 指令中可进行直接比较。

示例

图 2-17: STL 指令中 PLC 数据类型变量的比较



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

2.9.5 多重赋值 (V14 或更高版本)

优点

多重赋值能够对若干变量进行最优化编程 (例如, 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

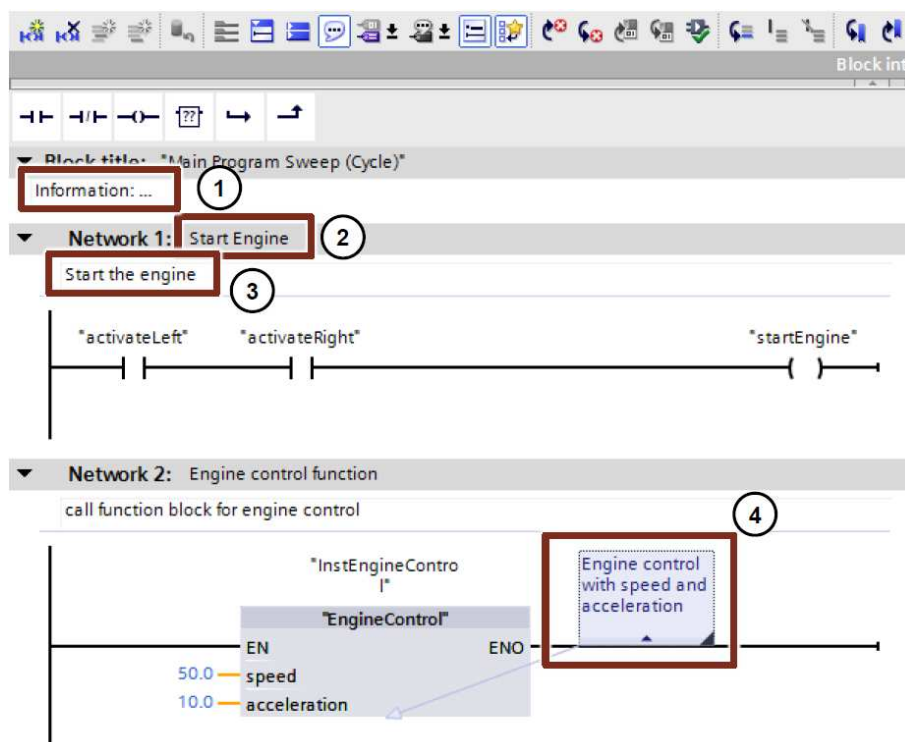
通过在程序中使用符号名称和注释，可以使您的同事易于理解和阅读代码。在下载至控制器的过程中，完整的符号与程序代码一并保存，因此，即使在没有离线项目可用的情况下，也可以快速维护设备。

建议

- 使用程序中的注释，以提高可读性。即使程序段被折叠，程序段标题注释也是可见的。
 - 设计程序代码时，要让同事也能直接理解程序。
- 在下面的示例中，可以看到在编辑器中注释程序的大量选项。

示例

在下图中，可以看到在 LAD 编辑器中注释的选项（FDB 有相同的功能）。
图 2-18：用户程序 (LAD) 中的注释



支持以下注释：

1. 块注释
2. 程序段标题注释
3. 程序段注释
4. 对指令、块和函数（打开、关闭等）进行注释。

在编程语言 SCL 和 STL 中，每行都可以用“//”进行注释。

示例

```
statFillingLevel := statRadius * statRadius * PI * statHight;  
// 计算中型罐的液位
```

注

有关更多信息，请参考以下条目：

在 STEP 7 (TIA Portal)，为什么在块编辑器中打开项目后不再显示已显示的文本、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

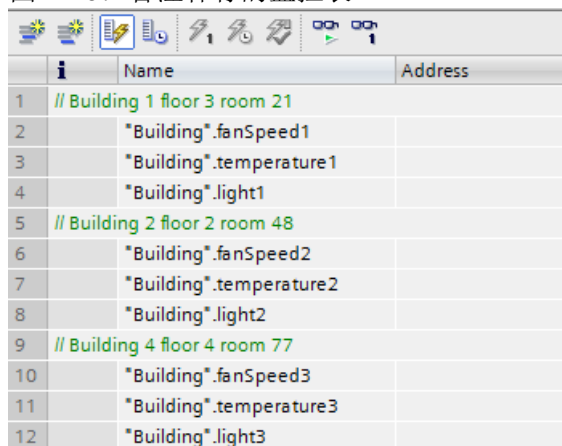
- 为了更好地组织，可以在监控表中创建注释行。

建议

- 始终使用注释行并细分您的监控表。
- 也可对具体变量进行注释。

示例

图 2-19：含注释行的监控表



The screenshot shows a monitoring table with 12 rows. The table has three columns: 'Name' and 'Address'. The rows are grouped into three sections, each starting with a comment row (row 1, 5, and 9). The comment rows are highlighted in green. The data rows contain variable names like '*Building*.fanSpeed1', '*Building*.temperature1', and '*Building*.light1'.

	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，通过逻辑地址或诊断地址来识别硬件和软件组件。

对于 S7-1200/1500，通过系统常量进行识别。S7-1200/1500 控制器的所有硬件和软件组件（如接口、模块、OB、...）均具有自己的系统常量。系统常量是在为集中式和分布式 I/O 设置设备配置的过程中自动创建的。

优点

- 可以通过模块名称而不是硬件标识来寻址。

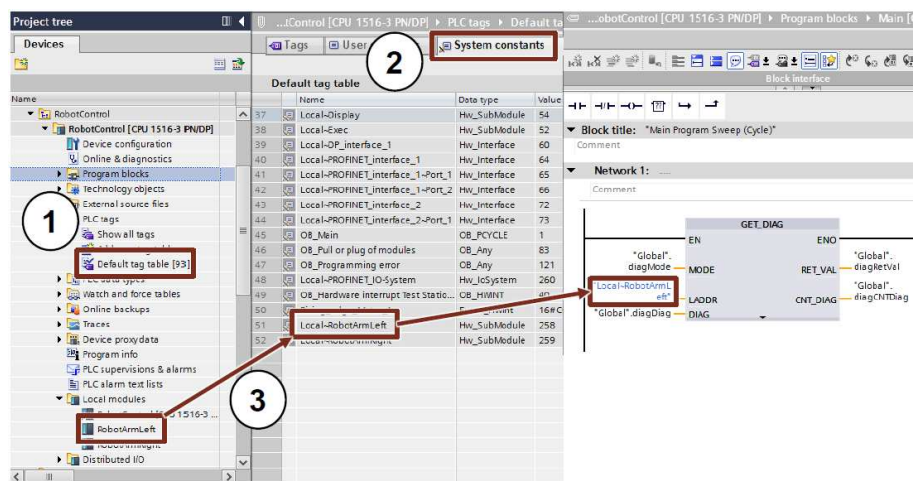
建议

- 分配与功能相关的模块名称，以便在编程过程中轻松识别模块。

示例

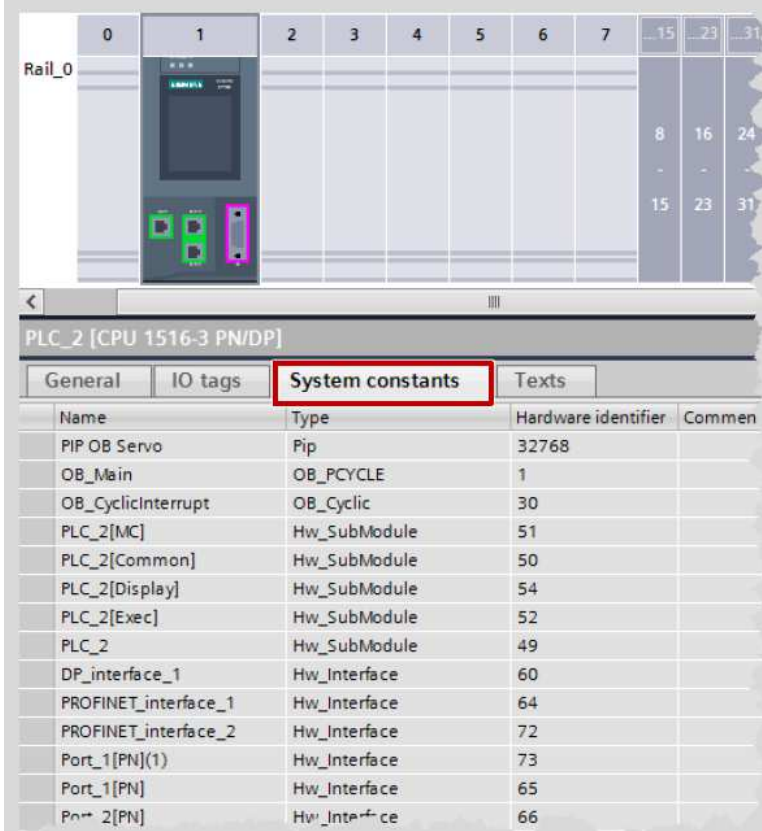
在下面示例中，可以看到如何在用户程序中使用系统常量。

图 2-20：用户程序中的“系统常量”。



1. 控制器的系统常量位于“PLC 变量 – 默认变量表”(PLC tags – Default tag table) 文件夹中。
2. 系统常量位于“默认变量表”中的一个单独的选项卡上。
3. 在本例中，为一个数字量输入模块分配了符号名称“RobotArmLeft”。具有此名称的模块位于系统常量表中。在用户程序中，“RobotArmLeft”与“GET_DIAG”诊断块连接。

注 打开“设备组态”(Device configuration), 可快速找到每个设备的系统常量。



PLC_2 [CPU 1516-3 PN/DP]				
General	IO tags	System constants	Texts	
Name	Type	Hardware identifier	Commen	
PIP_OB_Servo	Pip	32768		
OB_Main	OB_PCYLE	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 更多信息, 参见以下条目:
在 STEP 7 (TIA Portal) 中, 系统常量对 S7-1200/1500 有何意义?
<https://support.industry.siemens.com/cs/ww/en/view/78782835>

2.12 用户常量

使用用户常量，可以保存常量值。通常有用于组织块、函数和函数块的局部常量和用于控制器中整个用户程序的全局常量。

优点

- 用户常量可用于全局或局部更改所有使用位置的常量值。
- 通过用户常量，程序可变得更易读。

属性

- 局部用户常量是在块接口中定义的。
- 全局用户常量是在“PLC 变量”中定义的。
- 用户程序仅能对用户常量执行读访问。
- 对于受专有技术保护的块，用户常量不可见。

建议

- 使用用户常量可提高程序的可读性并可集中更改以下内容：
 - 错误代码
 - CASE 指令
 - 转换因子
 - 自然常量

示例

图 2-21: CASE 指令的块的局部用户常量

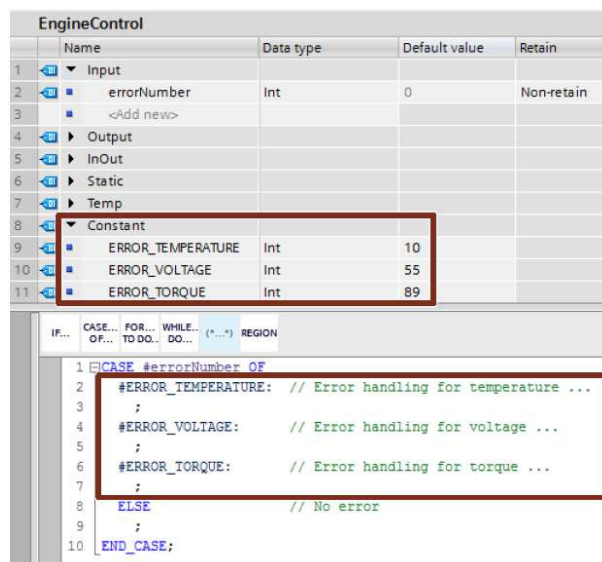
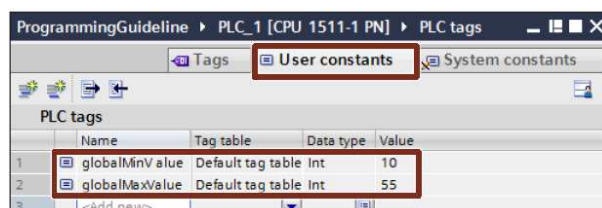


图 2-22: 控制器的全局用户常量



注 以下 FAQ 文章中提供了常量的其它应用情况：
 如何在 STEP 7 (TIA Portal) 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 变量的内部引用 ID

STEP 7、WinCC、Startdrive、Safety 和其它程序已集成到 TIA Portal 工程平台中。用户程序中的所有位置都会自动接受数据更改，而不管更改是发生在控制器中，还是发生在面板或驱动器上。因此，不会发生数据不一致的情况。

如果用户创建变量，则 TIA Portal 会自动创建一个唯一引用 ID。用户无法查看该引用 ID 或对其编程。此过程就是内部引用。在更改变量（地址）时，引用 ID 保持不变。

下图显示了数据的内部引用。

图 2-23: PLC 和 HMI 的内部引用 ID

PLC1				HMI1		
PLC Symbol name	Absolute address	Internal PLC reference ID	Internal HMI Reference ID	HMI Symbol name	Access mode	Connection with PLC
motor1	I0.0	000123	009876	motor1	<symbolic access>	PLC1_HMI1
valve2	Q0.3	000138	000578	valve2	<symbolic access>	PLC1_HMI1

注 该 ID 可通过以下方式更改：

- 重命名变量
- 更改类型
- 删除变量。

优点

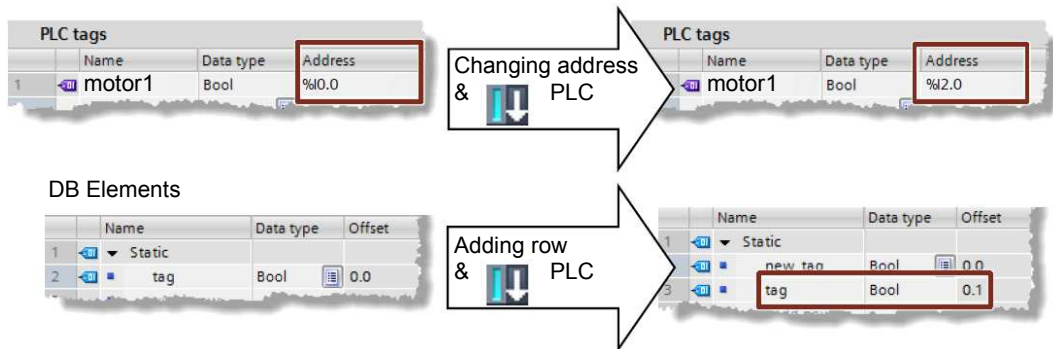
- 可以重新连接变量而不会改变内部关系。控制器、人机界面和驱动器之间的通信保持不变。
- 符号名称的长度对控制器与人机界面之间的通信负载没有影响。

属性

由于系统也会使用引用 ID 在内部寻址，如果更改 PLC 变量的地址，那么只需重新加载控制器。不必重新加载 HMI 设备（参见图 2-24：更改地址或添加行）。

图 2-24: 更改地址或添加行

PLC 变量



2.14 发生错误时的 STOP 模式

与 S7-300/400 不同的是，S7-1200/1500 中导致“STOP”模式的条件较少。由于 TIA Portal 中的一致性检查发生了改变，在大多数情况下，可以事先将 S7-1200/1500 控制器的“STOP”模式排除在外。在 TIA Portal 中编译时，已检查了程序块的一致性。与以前的控制器相比，这种方法让 S7-1200/1500 控制器的容错性更高。

优点

仅有三种故障情况会让 S7-1200/1500 控制器进入 STOP 模式。这样，错误管理编程就会变得更清楚和更容易。

属性

表 2-18: 对 S7-1200/1500 错误的响应

	错误	S7-1200	S7-1500
1.	超出循环监控时间一次	RUN	STOP (未组态 OB80 时)
2.	超出循环监控时间两次	STOP	STOP
3.	编程错误	RUN	STOP (未组态 OB121 时)

错误 OB:

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

另外，对于每个错误，还会在诊断缓冲区中自动创建一个条目。

注

对于 S7-1200/1500 控制器，还有其它可编程错误 OB（诊断错误、模块机架故障等）。

TIA Portal 在线帮助中的“事件和组织块”下面提供了有关 S7-1200/1500 的错误响应的详细信息。

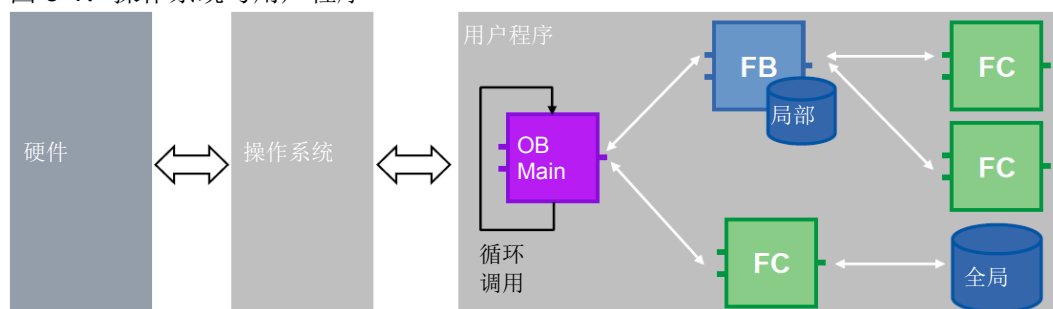
3 常规编程

3.1 操作系统与用户程序

SIMATIC 控制器包含操作系统和用户程序。

- 操作系统对未与特定控制任务（如重新启动的处理、过程映像更新、用户程序调用、错误处理、存储器管理等）关联的所有控制器功能和序列进行组织管理。操作系统是控制器不可缺少的组成部分。
- 用户程序包含处理特定自动化任务所需的全部程序块。用户程序通过程序块来编程并加载的控制器中。

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 OP 已存在于“程序块”(Program blocks) 文件夹中。该块由控制器进行处理，并在一个无限循环中重新调用。

3.2 程序块

STEP 7 (TIA Portal) 中具有来自以前 STEP 7 版本的完全类似的块类型：

- 组织块
- 函数块
- 函数
- 数据块

有经验的 STEP 7 用户可立即掌握这些块的使用，新用户也很容易熟悉编程。

优点

- 使用不同的块类型，可让程序的结构清晰有序。
- 由于程序结构安排有序，可以在项目中多次重新使用许多功能单元，并且也可在其它项目中使用。这些功能单元通常仅在组态上有所不同（参见 [3.2.9 块的可复用性](#)）。

- 用户的项目或工厂变得更加透明。即，可以更方便地检测、分析和消除工厂中的错误状态。工厂的维护变得更方便，编程中的错误也容易消除。

建议

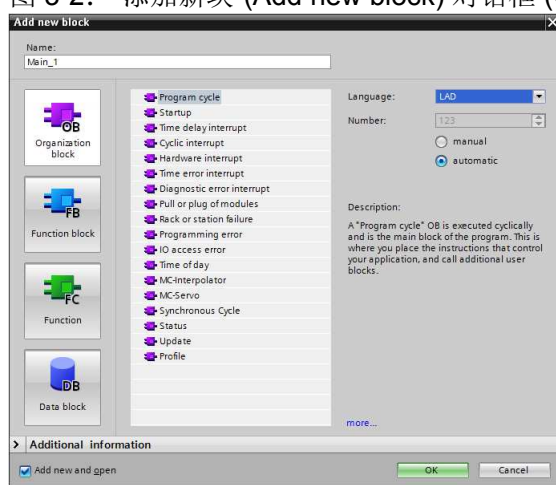
- 对自动化任务进行结构安排。
 - 将工厂的整个功能划分为多个具体区域，并形成子功能单元。再将这些功能单元划分为更小的单元与功能。一致划分至可通过不同的参数多次使用的功能。
 - 指定各功能单元之间的接口。定义需要由“外部公司”交付的功能的唯一接口。
- 所有组织块、函数块和函数都可用以下语言来编程：

表 3-1: 编程语言

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

3.2.1 组织块 (OB)

图 3-2: “添加新块”(Add new block) 对话框 (OB)



组织块 (OB) 是操作系统和用户程序之间的接口。组织块由操作系统调用，可以控制（例如）下列操作：

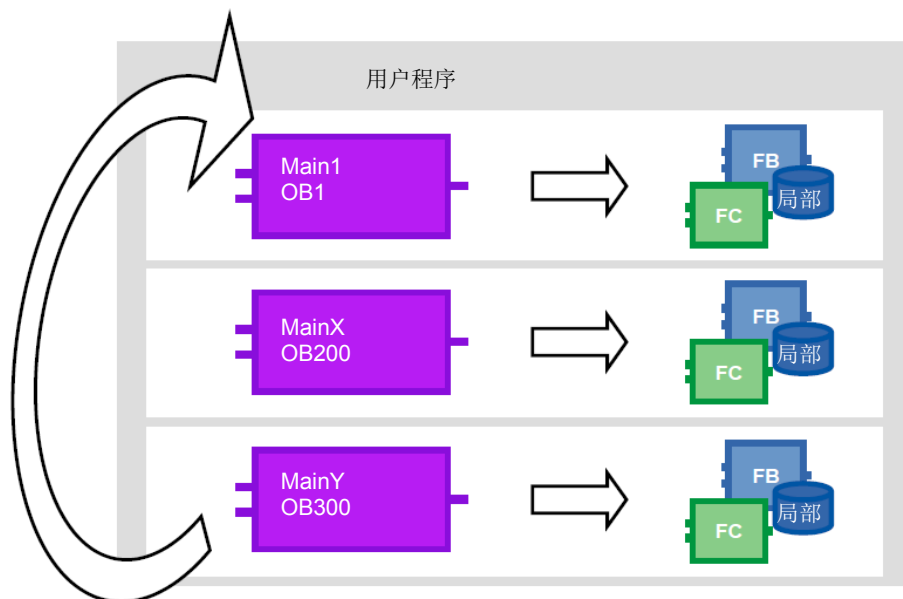
- 控制器的启动行为
- 循环程序执行
- 有中断控制的程序处理
- 错误处理

根据具体控制器，提供了不同 OB 类型。

属性

- 组织块由控制器的操作系统来调用。
- 在一个程序中可创建多个 Main OB。通过 OB 编号对 OB 进行顺序处理。

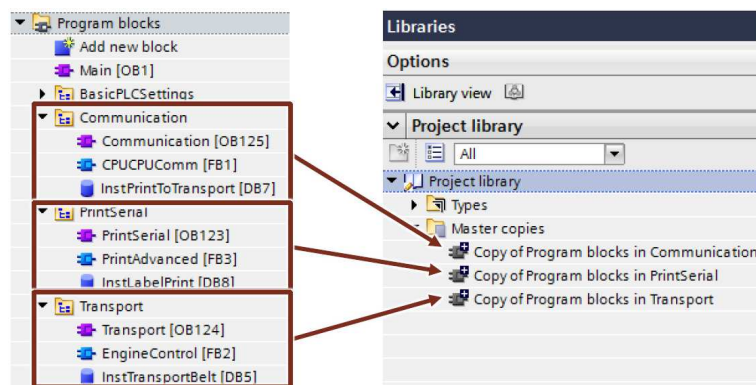
图 3-3: 使用多个 Main OB



建议

- 将可在控制器之间替换的不同程序部分封装为多个 Main OB。
- 避免不同 Main OB 之间的通信。随后它们可以彼此独立地使用。不过，如果要在具体的 Main OB 之间交换数据，请使用全局组织块（参见 4.2 不使用位存储器而使用全局数据块）。
- 将相互属于的所有程序部分划分到多个文件夹中，然后进行存储以便在项目中或全局库中再次使用。

图 3-4: 将各程序部分按顺序存储在项目库中



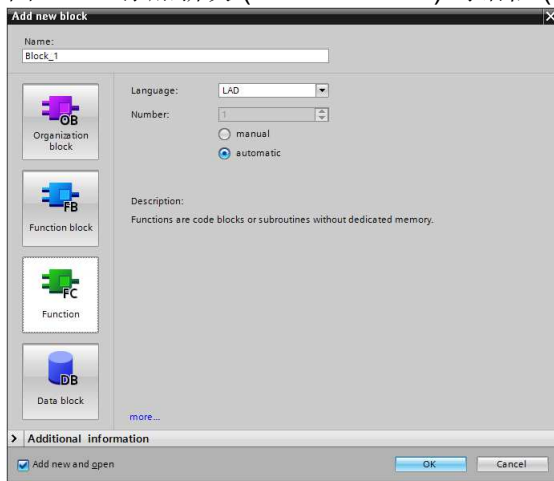
详细信息，参见 3.7 库。

注

更多信息，参见以下条目：
哪些组织块可在 STEP 7 (TIA Portal) 中使用？
<https://support.industry.siemens.com/cs/ww/en/view/40654862>

3.2.2 函数 (FC)

图 3-5: “添加新块”(Add new block) 对话框 (FC)



函数 (FC) 就是不带循环数据存储的块。这就是在下次调用之前无法保存块参数值的原因，在调用时，必须为其提供实际参数。

属性

- 函数 (FC) 就是不带循环数据存储的块。
- 在未优化的块中调用时，未定义临时变量。在优化的块中，值总是预设为默认值 (S7-1500 和 S7-1200 固件 V4 及以上版本)。这样，所得的行为并非偶然，而是可重复的。
- 为了永久保存函数的数据，可使用全局数据块的函数。
- 函数可具有多个输出。
- 函数值可直接在 SCL 的公式中再次使用。

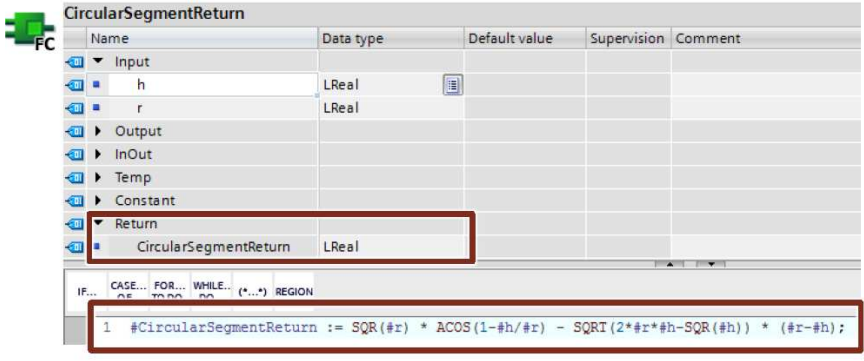
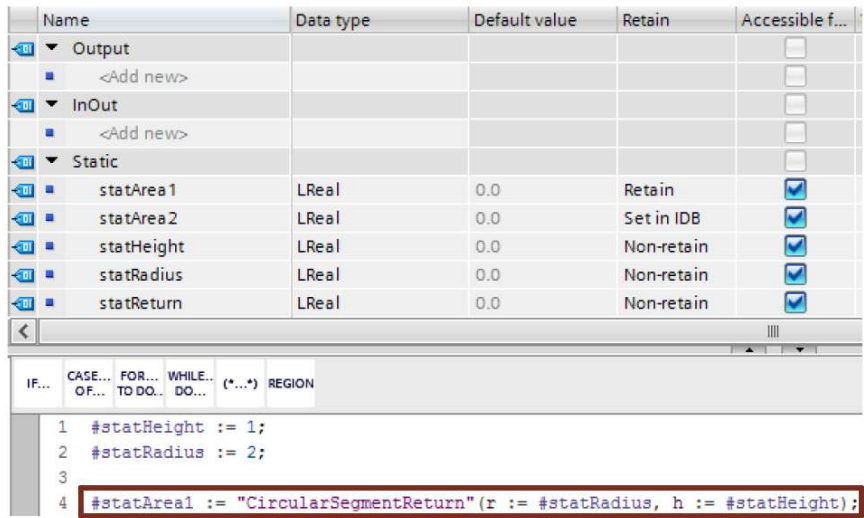
建议

- 将函数用于将在用户程序的不同位置经常调用的连续循环的应用程序。
- 使用该选项直接在 SCL 中再次使用函数值。
<操作数> := <函数名> (参数表);

示例

在下面的示例中，将在 FC 中编程一个数学公式。计算结果被直接声明为返回值，函数值可直接再次使用。

表 3-2: 再次使用函数值

步骤	说明
1.	<p>使用数学公式创建一个函数（循环段），并将“Return”值定义为公式的结果。</p> 
2.	<p>在任何块 (SCL) 中，通过循环段计算，调用该函数 FC。 <操作数> := <函数名> (参数表);</p> 

注

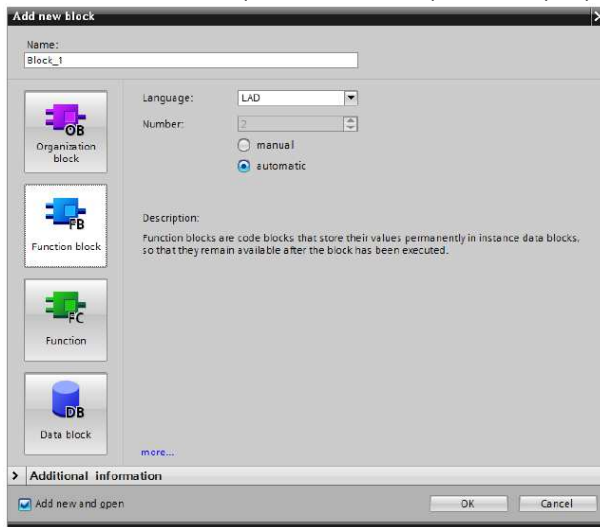
更多信息，参见以下条目：

在 STEP 7 (TIA Portal) 中，可以为 S7-1200/S7-1500 CPU 中的函数最多定义多少个参数？

<https://support.industry.siemens.com/cs/ww/en/view/99412890>

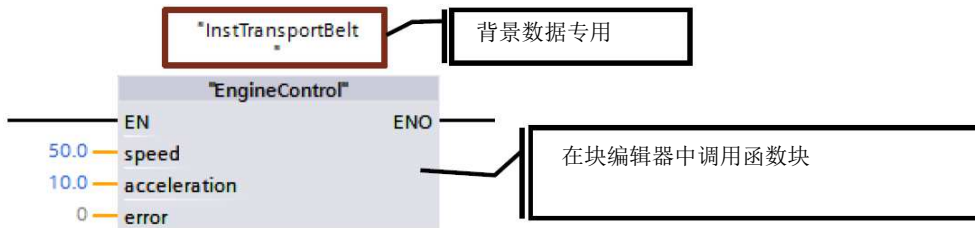
3.2.3 函数块 (FB)

图 3-6: “添加新块”(Add new block) 对话框 (FB)



函数块 (FB) 就是带有循环数据存储区的块，值将永久存储在存储区内。循环数据存储区是在背景数据块中实现的。

图 3-7: 调用函数块



属性

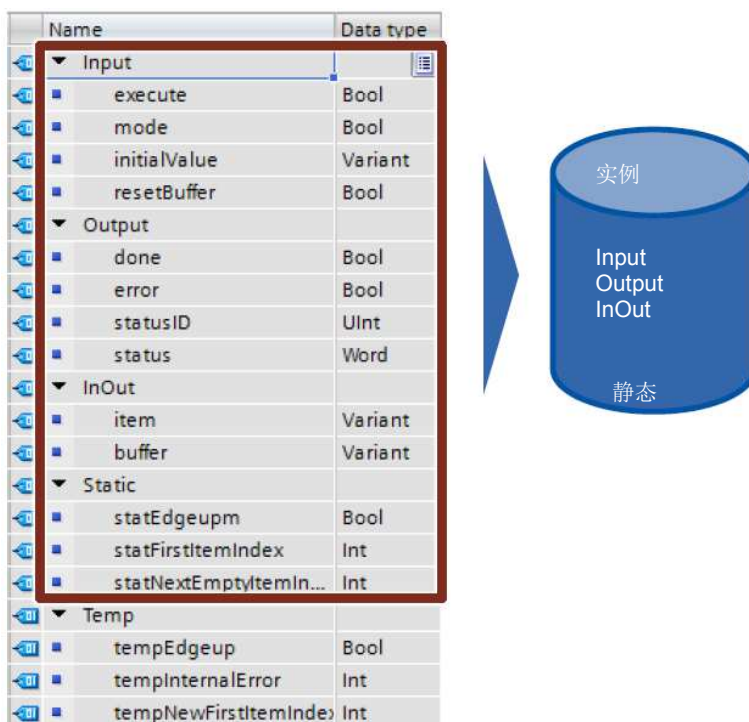
- 函数块 (FB) 就是带有循环数据存储区的块。
- 在未优化的块中调用时，未定义临时变量。在优化的块中，值总是预设为默认值（S7-1500 和 S7-1200 固件 V4）。这样，所得的行为并非偶然，而是可重复的。
- 静态变量在各循环之间保留其值。

建议

- 使用函数块来创建子程序并对用户程序进行结构安排。一个函数块可以在用户程序的不同位置被调用多次。这样，经常重复的程序部分的编程就变得更容易。
- 如果在用户程序中多次使用函数块，请使用单独的实例，最后是多实例。

3.2.4 实例

函数块的调用称为“实例”。实例所使用的数据保存在背景数据块中。背景数据块总是根据 FB 接口的规格来创建，因此不能在背景数据块中进行更改。
图 3-8：函数块接口的结构



背景数据块包含一个永久存储器，其带有接口 **Input**、**Output**、**InOut** 和 **Static**。易失性存储器（L 堆栈）中存储临时变量。L 堆栈总是仅对当前处理有效。也就是说，必须在每个循环中对临时变量进行初始化。

属性

- 背景数据块总是分配给函数块。
- 不必在 TIA Portal 中手动创建背景数据块，它们是在调用函数块时自动创建的。
- 背景数据块的结构在相应函数块中指定，并且只能在该函数块中来更改。

建议

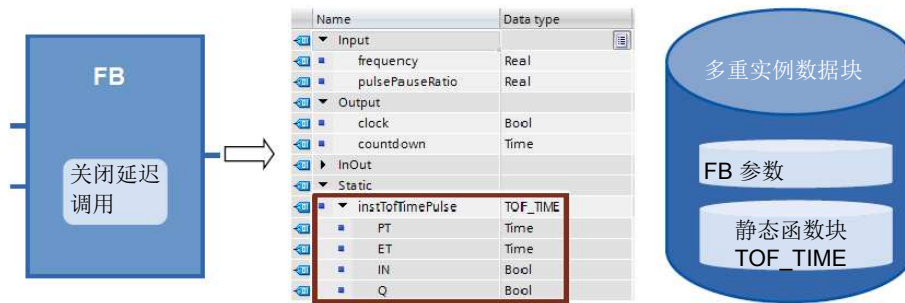
- 编程时，要让背景数据块的数据只能由相应函数块来更改。这样就可以保证该块可在各种类型的项目中通用。

有关详细信息，请参见 [3.4.1 通过块接口进行数据交换](#)。

3.2.5 多重实例

通过多重实例，被调用的函数块可将其数据存储在被调用的函数块的背景数据块中。也就是说，如果在一个函数块中调用其它函数块，那么它会将其数据保存在上层函数块的背景数据块中。因此，即时被调用的块已转移，其功能仍保留。

下图显示了一个使用其它函数块（“IEC Timer”）的函数块。所有数据都保存在一个多重实例数据块中。因此可以创建具有独立时间行为的块，如时钟发生器。



优点

- 可重复使用
- 可多次调用
- 背景数据块更少，程序更清晰
- 简单复制程序
- 具有在编程期间进行结构安排的便利选项

属性

- 多重实例是背景数据块内的存储区域。

建议

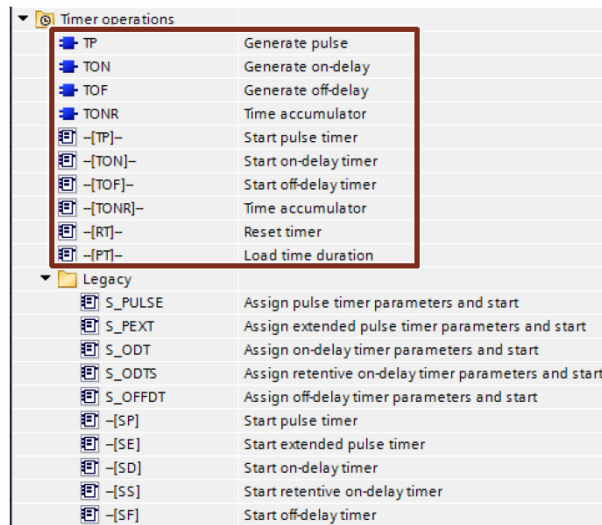
可使用多重实例来：

- 减少背景数据块的数量
- 创建清晰且可重复使用的用户程序
- 程序布局功能，如定时器、计数器、边沿检测。

示例

如果需要时间和计数器功能，请使用“IEC Timer”块和“IEC Counter”块，而不使用绝对寻址的 SIMATIC 定时器。如果可能，总是要在这里使用多重实例。这会在用户程序中保持较低的块数量。

图 3-10: IEC 定时器的库



Timer operations	
TP	Generate pulse
TON	Generate on-delay
TOF	Generate off-delay
TONR	Time accumulator
-(TP)-	Start pulse timer
-(TON)-	Start on-delay timer
-(TOF)-	Start off-delay timer
-(TONR)-	Time accumulator
-(RT)-	Reset timer
-(PT)-	Load time duration
Legacy	
S_PULSE	Assign pulse timer parameters and start
S_PEXT	Assign extended pulse timer parameters and start
S_ODT	Assign on-delay timer parameters and start
S_ODTS	Assign retentive on-delay timer parameters and start
S_OFFDT	Assign off-delay timer parameters and start
-(SP)	Start pulse timer
-(SE)	Start extended pulse timer
-(SD)	Start on-delay timer
-(SS)	Start retentive on-delay timer
-(SF)	Start off-delay timer

注

更多信息，参见以下条目：
在 STEP 7 (TIA Portal) 中，如何声明 S7-1500 的定时器和计数器？
<https://support.industry.siemens.com/cs/ww/en/view/67585220>

3.2.6 以参数形式传送实例 (V14)

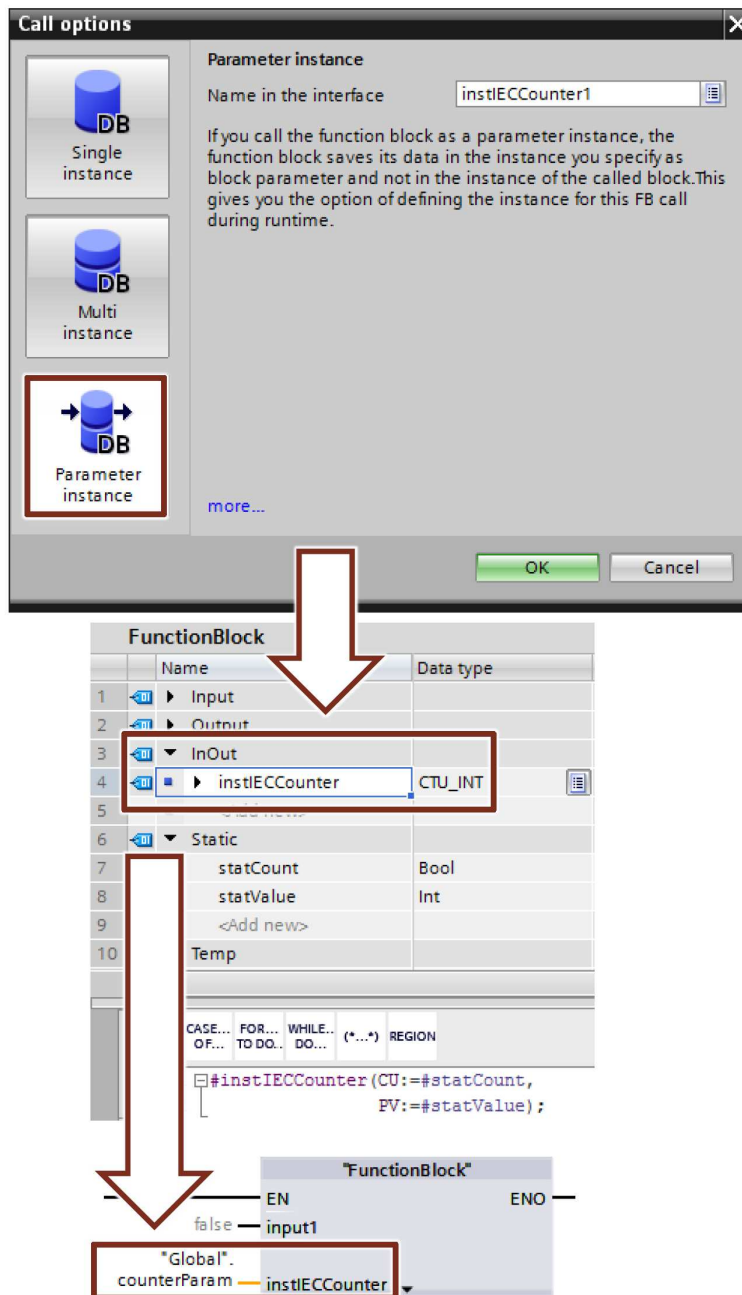
可以将被调用块的实例定义为 InOut 参数。

优点

- 可以创建其动态实例已传送的标准化函数。
- 仅在调用该块时才指定使用哪个实例。

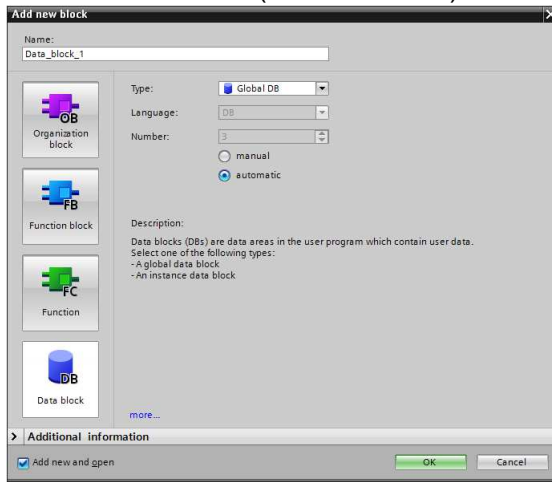
示例

图 3-11：以参数形式传送实例



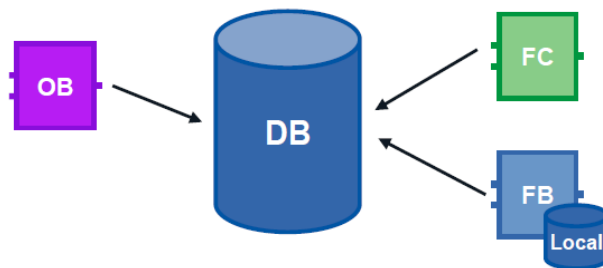
3.2.7 全局数据块 (DB)

图 3-12: “添加新块”(Add new block) 对话框 (DB)



变量数据位于提供给整个用于程序的数据块中。

图 3-13: 全局数据块作为中央数据存储



优点

- 存储器区域结构有序
- 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由所有数据类型任意构成。
- 全局数据块可通过程序编辑器来创建，或根据先前创建的“用户定义 PLC 数据类型”来创建（参见 [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

- 当数据在不同程序部分或块中使用，请使用全局数据块。

注

更多信息，参见以下条目：

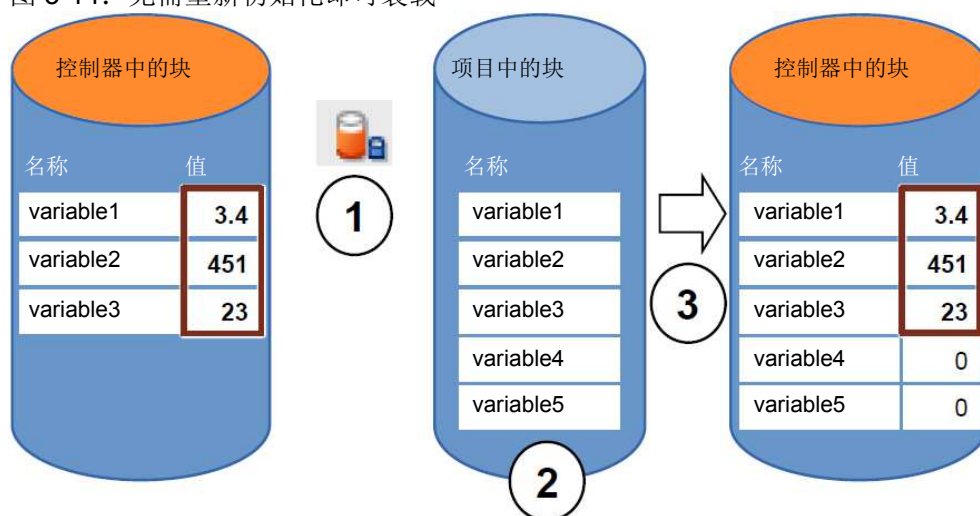
如何在 STEP 7 (TIA Portal) 中结构化全局数据块的声明表？

<https://support.industry.siemens.com/cs/ww/en/view/68015630>

3.2.8 无需重新初始化即可下载

为了更改已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期间扩展优化函数或数据块的接口的选项。用户可以加载更改后的块，而无需将控制器置于 STOP 模式，也不会影响已加载的变量的实际值。

图 3-14：无需重新初始化即可装载



当控制器处于 RUN 模式时，请执行以下步骤。

1. 启用“下载但不进行初始化”(Downloading without reinitialization)
2. 在现有块中插入新定义的变量
3. 将块加载到控制器中

优点

- 重新加载新定义的变量而不会中断运行过程。控制器保留在“RUN”模式。

属性

- 只有优化的块才能实现下载而不进行初始化。
- 将会初始化新定义的变量。其余变量保留其当前值。
- 具有预留空间的块需要控制器中的更多存储器空间。
- 存储器预留空间取决于控制器的工作存储器，但最大为 2 MB。
- 假设已为块定义了存储器预留空间。
- 默认情况下，存储器预留空间设置为 100 字节。
- 将分别为每个块定义存储器预留空间。
- 块可以扩展。

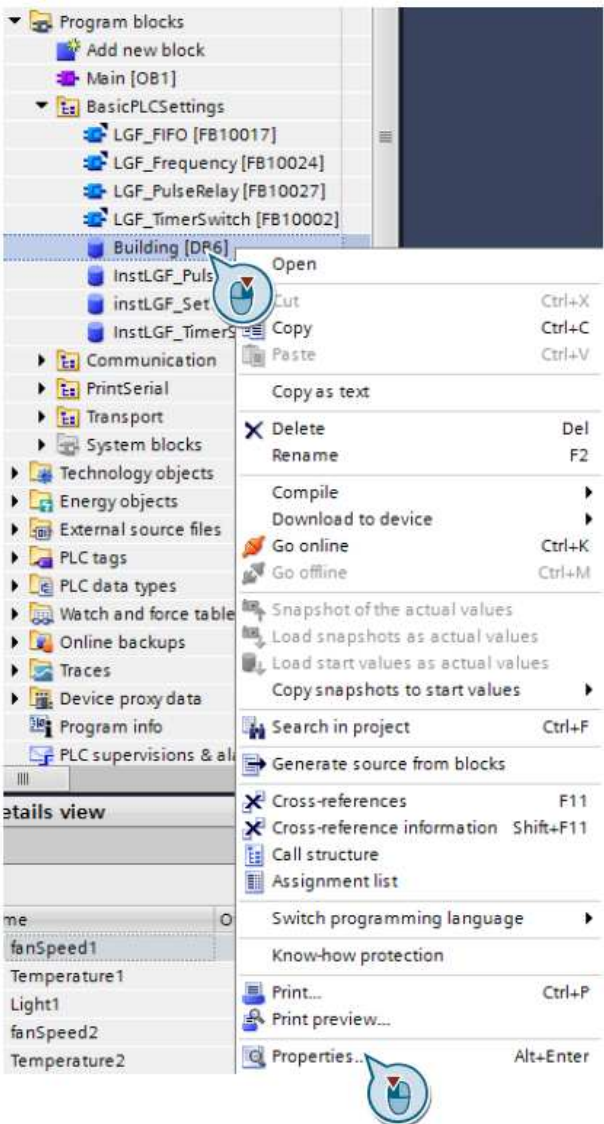
建议

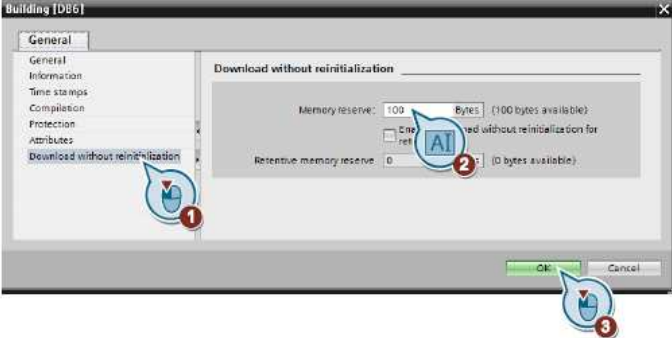
- 在调试期间，为需要扩展的块定义存储器预留空间（例如，测试块）。调试过程不会受到下载的干扰，因为现有变量的实际值仍然存在。

示例：在块上设置存储器预留空间

下表说明了如何针对下载而不进行初始化的情况设置存储器预留空间。

表 3-3: 设置存储器预留空间

步骤	说明
1.	<p>右键单击项目树中任意一个优化的块，然后选择“属性”(Properties)。</p> 

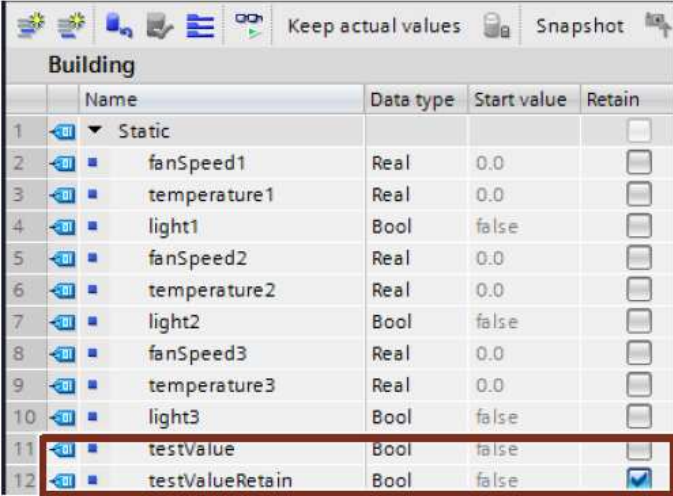
步骤	说明
2.	 <p>单击“下载但不进行初始化”(Download without reinitialization)。</p> <p>3. 在“存储器预留空间”(Memory reserve) 输入所需的存储器预留空间。</p> <p>4. 单击“OK”确认。</p>

注 也可以在 TIA portal 中设置新块的存储器预留空间默认值。在菜单栏中，导航至“选项 – 设置”(Options – Settings)，然后来到“PLC 编程 – 常规 – 下载而不进行初始化”(PLC programming General – Download without reinitialization)。

示例：无需重新初始化即可下载

以下示例显示了如何实现无需重新初始化即可下载。
表 3-4 无需重新初始化即可下载

步骤	说明																																								
1.	前提：必须设置存储器预留空间（见上面）。																																								
2.	例如，打开一个优化的全局数据块。																																								
3.	<p>单击“激活存储器预留空间”(Activate memory reserve) 按钮，然后按“确定”(OK) 确认该对话框。</p>  <table border="1"> <thead> <tr> <th></th> <th>Name</th> <th>Data type</th> <th>Start</th> <th>Retain</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Static</td> <td></td> <td></td> <td></td> </tr> <tr> <td>2</td> <td>fanSpeed1</td> <td>Real</td> <td>0.0</td> <td><input type="checkbox"/></td> </tr> <tr> <td>3</td> <td>temperature1</td> <td>Real</td> <td>0.0</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>4</td> <td>light1</td> <td>Bool</td> <td>false</td> <td><input type="checkbox"/></td> </tr> <tr> <td>5</td> <td>fanSpeed2</td> <td>Real</td> <td>0.0</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>6</td> <td>temperature2</td> <td>Real</td> <td>0.0</td> <td><input checked="" type="checkbox"/></td> </tr> <tr> <td>7</td> <td>light2</td> <td>Bool</td> <td>false</td> <td><input checked="" type="checkbox"/></td> </tr> </tbody> </table>		Name	Data type	Start	Retain	1	Static				2	fanSpeed1	Real	0.0	<input type="checkbox"/>	3	temperature1	Real	0.0	<input checked="" type="checkbox"/>	4	light1	Bool	false	<input type="checkbox"/>	5	fanSpeed2	Real	0.0	<input checked="" type="checkbox"/>	6	temperature2	Real	0.0	<input checked="" type="checkbox"/>	7	light2	Bool	false	<input checked="" type="checkbox"/>
	Name	Data type	Start	Retain																																					
1	Static																																								
2	fanSpeed1	Real	0.0	<input type="checkbox"/>																																					
3	temperature1	Real	0.0	<input checked="" type="checkbox"/>																																					
4	light1	Bool	false	<input type="checkbox"/>																																					
5	fanSpeed2	Real	0.0	<input checked="" type="checkbox"/>																																					
6	temperature2	Real	0.0	<input checked="" type="checkbox"/>																																					
7	light2	Bool	false	<input checked="" type="checkbox"/>																																					

步骤	说明																																																																						
4.	添加新变量（也可以是保持性变量）。  <table border="1" data-bbox="469 315 1145 808"> <thead> <tr> <th colspan="5">Building</th> </tr> <tr> <th></th> <th>Name</th> <th>Data type</th> <th>Start value</th> <th>Retain</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Static</td> <td></td> <td></td> <td><input type="checkbox"/></td> </tr> <tr> <td>2</td> <td>fanSpeed1</td> <td>Real</td> <td>0.0</td> <td><input type="checkbox"/></td> </tr> <tr> <td>3</td> <td>temperature1</td> <td>Real</td> <td>0.0</td> <td><input type="checkbox"/></td> </tr> <tr> <td>4</td> <td>light1</td> <td>Bool</td> <td>false</td> <td><input type="checkbox"/></td> </tr> <tr> <td>5</td> <td>fanSpeed2</td> <td>Real</td> <td>0.0</td> <td><input type="checkbox"/></td> </tr> <tr> <td>6</td> <td>temperature2</td> <td>Real</td> <td>0.0</td> <td><input type="checkbox"/></td> </tr> <tr> <td>7</td> <td>light2</td> <td>Bool</td> <td>false</td> <td><input type="checkbox"/></td> </tr> <tr> <td>8</td> <td>fanSpeed3</td> <td>Real</td> <td>0.0</td> <td><input type="checkbox"/></td> </tr> <tr> <td>9</td> <td>temperature3</td> <td>Real</td> <td>0.0</td> <td><input type="checkbox"/></td> </tr> <tr> <td>10</td> <td>light3</td> <td>Bool</td> <td>false</td> <td><input type="checkbox"/></td> </tr> <tr> <td>11</td> <td>testValue</td> <td>Bool</td> <td>false</td> <td><input type="checkbox"/></td> </tr> <tr> <td>12</td> <td>testValueRetain</td> <td>Bool</td> <td>false</td> <td><input checked="" type="checkbox"/></td> </tr> </tbody> </table>	Building						Name	Data type	Start value	Retain	1	Static			<input type="checkbox"/>	2	fanSpeed1	Real	0.0	<input type="checkbox"/>	3	temperature1	Real	0.0	<input type="checkbox"/>	4	light1	Bool	false	<input type="checkbox"/>	5	fanSpeed2	Real	0.0	<input type="checkbox"/>	6	temperature2	Real	0.0	<input type="checkbox"/>	7	light2	Bool	false	<input type="checkbox"/>	8	fanSpeed3	Real	0.0	<input type="checkbox"/>	9	temperature3	Real	0.0	<input type="checkbox"/>	10	light3	Bool	false	<input type="checkbox"/>	11	testValue	Bool	false	<input type="checkbox"/>	12	testValueRetain	Bool	false	<input checked="" type="checkbox"/>
Building																																																																							
	Name	Data type	Start value	Retain																																																																			
1	Static			<input type="checkbox"/>																																																																			
2	fanSpeed1	Real	0.0	<input type="checkbox"/>																																																																			
3	temperature1	Real	0.0	<input type="checkbox"/>																																																																			
4	light1	Bool	false	<input type="checkbox"/>																																																																			
5	fanSpeed2	Real	0.0	<input type="checkbox"/>																																																																			
6	temperature2	Real	0.0	<input type="checkbox"/>																																																																			
7	light2	Bool	false	<input type="checkbox"/>																																																																			
8	fanSpeed3	Real	0.0	<input type="checkbox"/>																																																																			
9	temperature3	Real	0.0	<input type="checkbox"/>																																																																			
10	light3	Bool	false	<input type="checkbox"/>																																																																			
11	testValue	Bool	false	<input type="checkbox"/>																																																																			
12	testValueRetain	Bool	false	<input checked="" type="checkbox"/>																																																																			
5.	将该块下载到控制器。																																																																						
6.	结果： <ul style="list-style-type: none"> • 块的实际值保留 																																																																						

注

TIA Portal 在线帮助中的“加载块扩展但不进行初始化”下页提供了详细信息。有关更多信息，请参考以下条目：
 如何在 STEP 7-1500 (TIA Portal) 中结构化全局数据块的声明表？
<https://support.industry.siemens.com/cs/ww/en/view/68015630>

3.2.9 块的可再用性

块的概念提供了用于以结构化方式有效编程的众多选项。

优点

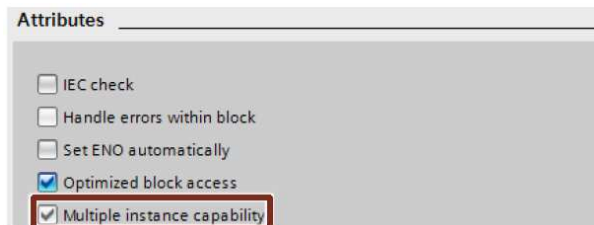
- 块可在用户程序的任何位置通用。
- 块可在不同项目中通用。
- 当每个块接收独立任务时，将会自动创建清晰、有序的用户程序。
- 错误源明显减少。
- 可实现简单诊断。

建议

如果要重复使用块，请注意以下建议：

- 总是要将块视为封装的函数。也就是说，每个块都代表整个用户程序中一个已完成的部分任务。
- 使用多个循环 **Main OB** 将各个工厂部分分组。
- 总是要通过块的接口（而不是通过其实例）在块之间执行数据交换（参见 [3.4.1 通过块接口进行数据交换](#)）。
- 请勿使用与项目特定相关的数据，并避免以下块操作：
 - 访问全局数据库和使用单实例数据块
 - 访问变量
 - 访问全局常量
- 可重复使用的块的要求与库中受专有技术保护的块相同。因此，必须基于“多实例功能”(Multiple instance capability) 块属性来检查块的可再用性。检查之前对块进行编译。

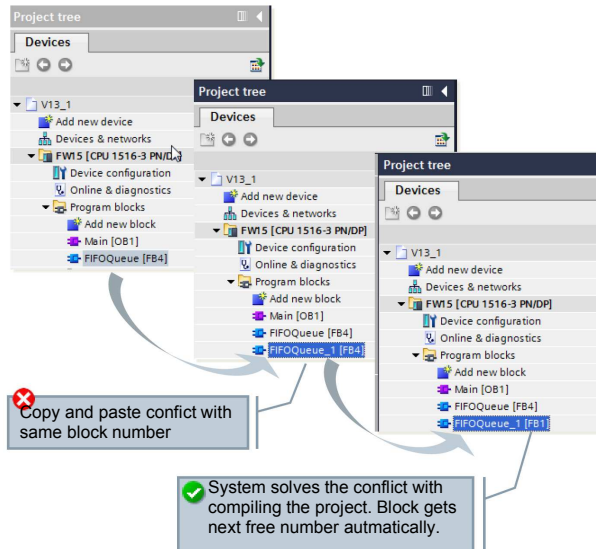
图 3-15: 块属性



3.2.10 块的自动编号

为了进行内部处理，系统将会分配所需的块编号（在块属性中设置）。

图 3-16: 块的自动编号



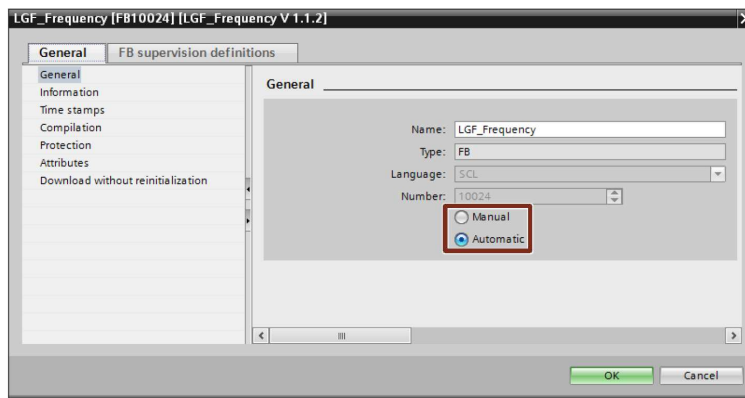
优点

- TIA Portal 会在编译期间自动删除冲突的块编号（例如，因复制产生的块编号）。

建议

- 保持现有设置“自动”(Automatic) 不变。

图 3-17: 在块中设置



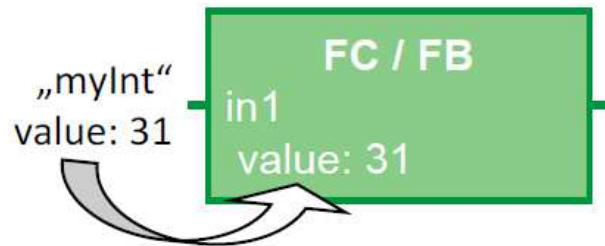
3.3 块接口类型

FB 和 FC 具有三种不同的接口类型：In、InOut 和 Out。通过这些接口类型，将向块提供参数。这些参数在该块中得到处理并再次输出。InOut 参数用于将数据传送到被调用的块以及返回结果。有两种不同的数据参数传送方式可供选择。

3.3.1 传值

调用块时，实际参数的值被复制到块的形式参数上。为此，在被调用的块中提供额外的存储器。

图 3-18: 数值传送



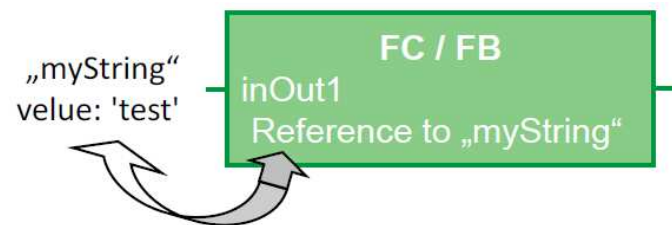
属性

- 每个块显示与传送参数相同的性能
- 调用块时复制值

3.3.2 传引用

调用块时，将引用传送到实际参数的地址。无需额外存储空间。

图 3-19: 引用实际参数（在该参数的数据存储之前）



属性

- 每个块显示与引用参数相同的性能。
- 调用块时引用实际参数，即通过访问，直接读取或写入实际参数的值。

建议

- 为了避免不必要地增加所需的数据存储空间，通常要将 InOut 接口类型用于结构化变量（例如，ARRAY、STRUCT、STRING 等类型的结构化变量）

3.3.3 参数传送一览

下表概述了具有基本或结构化数据类型的 S7-1200/1500 块参数的传送。

表 3-5: 参数传送一览

块类型/形式参数		基本数据类型	结构化数据类型
FC	Input	Copy	Reference
	Output	Copy	Reference
	InOut	Copy	Reference
FB	Input	Copy	Copy
	Output	Copy	Copy
	InOut	Copy	Reference

注 当调用块时传送具有“未优化访问”属性的优化数据时，其通常作为副本传送。当块包含许多结构化参数时，可能会很快导致块的临时存储区域（本地数据堆栈）溢出。可以通过为两个块设置相同的访问类型来避免这种情况（参见 [2.6.5 优化和未优化访问块之间的参数传递](#)）。

3.4 存储器概念

对于 STEP 7，通常有全局和局部存储区域间的差别。全局存储区域可供用户程序中的每个块使用。局部存储区域仅在相应的块中可用。

3.4.1 通过块接口进行数据交换

如果您要封装函数并仅通过接口来编程块之间的数据交换，则可以看到明显优点。

优点

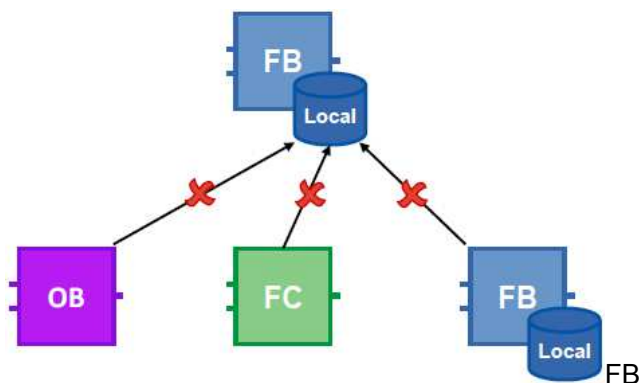
- 可以用部分任务的现成块，以模块化方式构成程序。
- 程序易于扩展和维护。
- 程序代码更易读，易测试，因为不存在隐藏的交叉访问。

建议

- 如果可能，请仅使用局部变量。因此，可以以模块化方式通用使用这些块。

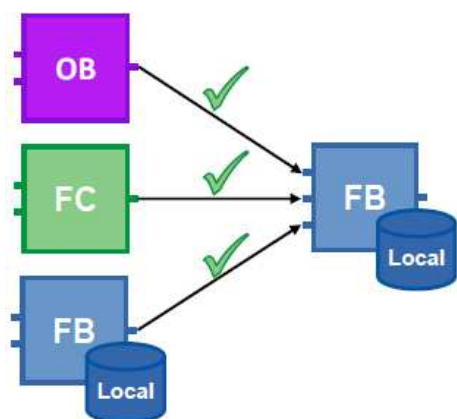
- 通过块接口（In、Out、InOut）实现数据交换，以确保块的可再用性。
- 仅将背景数据块用作相应函数块的局部存储器。不应将其它块写入背景数据块。

图 3-20: 避免访问背景数据块



如果仅将块接口用于数据交换，则可以确保相互独立地使用所有块。

图 3-21: 用于数据交换的块接口



3.4.2 全局存储器

当能够从用户程序的任何位置访问存储器时，将会全局调用这些存储器。具有依赖于硬件的存储器（如位存储器、定时器、计数器等）和全局数据块。对于依赖于硬件的存储器区域，由于这些区域可能已被使用，有可能无法将程序移植到任何控制器。因此，应使用全局数据块而不使用依赖于硬件的存储器区域。

优点

- 用户程序可普遍使用，不依赖于硬件。
- 可模块化组态用户程序，无需为不同用户将其划分为多个位存储器区域。
- 与出于兼容性原因而未优化的位存储器区域相比，优化的全局数据块明显功能更强大。

建议

- 请勿使用任何位存储器，而要使用全局数据块。
- 避免使用依赖于硬件的存储器，例如，时钟存储器或计数器。将 IEC 计数器和定时器与多重实例结合使用（参见 [3.2.5 多重实例](#)）。关于 IEC 定时器，请参见“指令 – 基本指令 – 定时器操作”。

图 3-22: IEC 定时器

Timer operations	
TP	Generate pulse
TON	Generate on-delay
TOF	Generate off-delay
TONR	Time accumulator
[TP]-	Start pulse timer
[TON]-	Start on-delay timer
[TOF]-	Start off-delay timer
[TONR]-	Time accumulator
[RT]-	Reset timer
[PT]-	Load time duration

3.4.3 局部存储器

- 静态变量
- 临时变量

建议

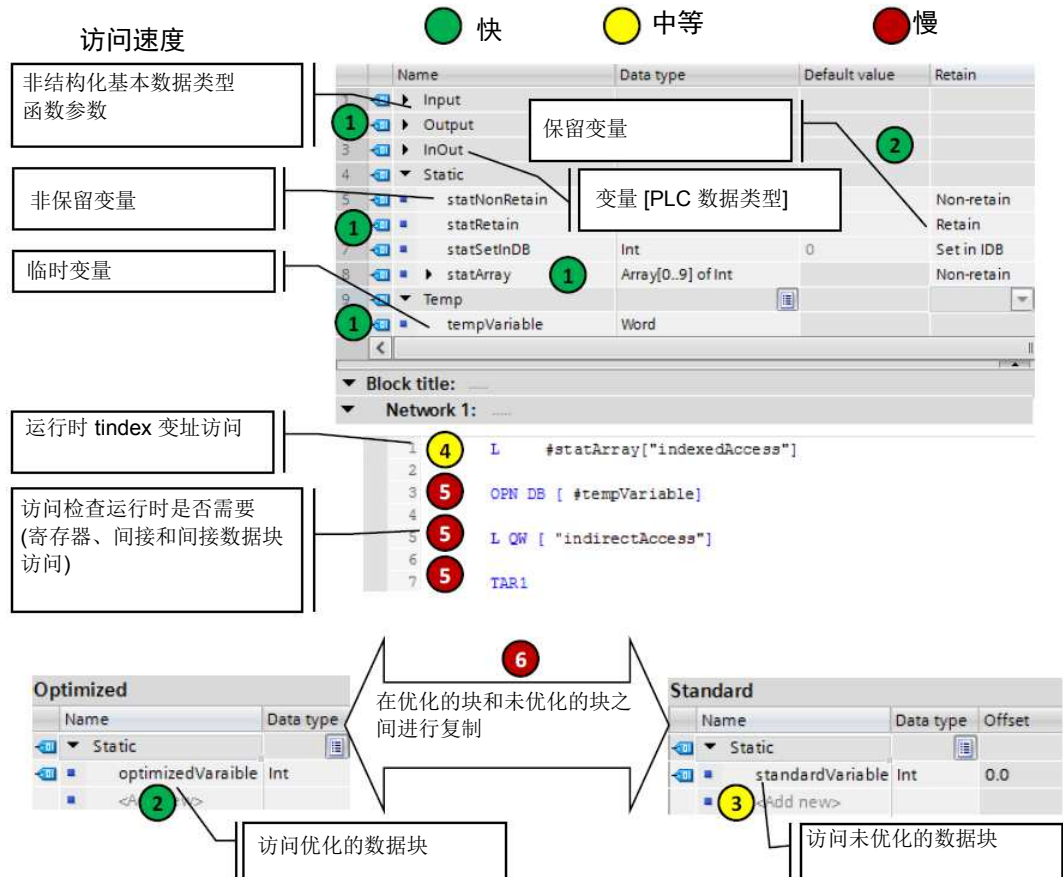
- 如果下一个循环需要该值，则可使用静态变量。
- 将临时变量用作当前循环中的缓存存储器。临时变量的存储时间短于静态变量的存储时间。
- 如果非常频繁地访问 Input/Output 变量，则使用临时变量作为中间存储器以节省运行时间。

注 优化的块：在任何块调用中，将使用默认值来初始化临时变量（S7-1500 / S7-1200 固件 V4 或更高版本）。
未优化的块：未针对块的每次调用定义临时变量。

3.4.4 存储器区域的访问速度

STEP 7 提供了不同的存储器区域访问方式。出于与系统相关的原因，对不同存储器区域的访问速度或快或慢。

图 3-23: 不同的存储器访问



在 S7-1200/1500 按降序高速访问

1. 优化的块：临时变量、函数和函数块的参数、非保持性静态变量、变量 [PLC数据类型]
2. 已知进行编译访问的优化块：
 - 保持性函数块变量
 - 优化的全局数据块
3. 访问未优化的块
4. 通过在运行时计算的索引（如 Motor [i]）进行变址访问
5. 需要在运行时检查的访问
 - 访问在运行时创建的数据块或间接打开的数据块（如 OPN DB[i]）
 - 寄存器访问或间接存储器访问
6. 在优化和未优化的块之间复制结构（字节数组除外）

3.5 保持性

在电源发生故障时，控制器将保持性数据与其缓冲能量一起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，将使用保持性数据恢复程序处理。根据具体控制器，保持性数据量的大小各不相同。

表 3-6: S7-1200/1500 的保持性存储器

控制器	可使用的保持性存储器，用于位存储器、定时器、计数器、数据块和工艺对象
CPU 1211C, 1212C, 1214C, 1215C, 1217C	10 kB
CPU 1511-1 PN	88 kB
CPU 1513-1 PN	88 kB
CPU 1515-2 PN, CPU 1516-3 PN/DP	472 kB
CPU 1518-4 PN/DP	768 kB

表 3-7: S7-1200 与 S7-1500 的差异

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

优点

- 保持性数据会在控制器转为 **STOP** 模式再回到 **RUN** 模式或者在发生电源故障或控制器重启的情况下保持其值。

属性

可以单独为优化数据块的基本变量设置保持性。未优化的数据块只能完全定义为保持性或非保持性。

保持性数据可通过“存储器复位”或“复位至出厂设置”操作来删除，方式为：

- 操作控制器上的开关 (MRES)
- 控制器的显示屏
- 通过 **STEP 7 (TIA Portal)** 在线进行

建议

- 不要使用设置“在背景数据块中设置”(Set in IDB)。总是在函数块中（而不是在背景数据块中）设置保持性数据。
“在背景数据块中设置”(Set in IDB) 设置会增加程序序列的处理时间。总是要针对函数块中的接口选择“非保留”(Non-retain) 或“保留”(Retain)。

图 3-24: 程序编辑器 (函数块接口)

Static				
instToTimePulse	TOF_...			Non-retain
instToTimePause	TOF_TIME			Non-retain
statFrequency	Real	0.0		Non-retain
statTimePeriod	Time	T#0ms		Non-retain
statTimePulse	Time	T#0ms		Non-retain

图 3-25: 程序编辑器 (数据块)

Building					
Name	Data type	Start value	Retain	Accessible f...	
Static					
fanSpeed1	Real	0.0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
temperature1	Real	0.0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
light1	Bool	false	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
fanSpeed2	Real	0.0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
temperature2	Real	0.0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
light2	Bool	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
fanSpeed3	Real	0.0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
temperature3	Real	0.0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
light3	Bool	false	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

示例: 保持性 PLC 变量

保持性数据的设置是在 PLC 变量、函数块和数据块的表中进行的。

图 3-26: 在 PLC 变量表中设置保持性变量

Retain memory dialog box settings:

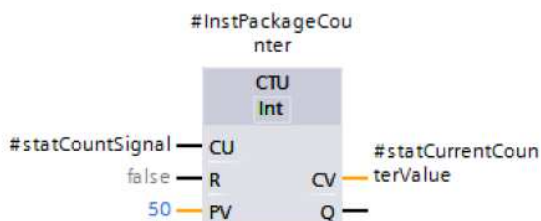
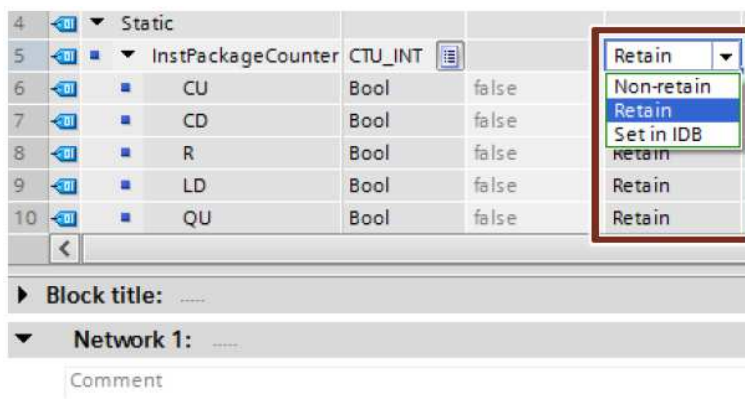
- Number of memory bytes starting at MB0: 0
- Number of SIMATIC timers starting at T0: 0
- Number of SIMATIC counters starting at C0: 0
- Currently available retain memory (bytes): 90784

Callout text: 从地址 0 开始 可以设置保持性! 例如从 MB0、T0 或 C0

示例：保持性计数器

也可以将函数的实例（定时器、计数器等）声明为保持性。如章节 [3.2.5 多重实例](#) 中所述，应始终将函数等编程为多重实例。

图 3-27：保持性计数器作为多重实例



注 如果 PLC 上的保持性存储器容量不充分，则可以数据块的形式存储数据，这些数据块仅位于 PLC 的装载存储器中。下面的文章以 S7-1200 为例进行了说明。本编程就能适用于 S7-1500。
更多信息，参见以下条目：
在 STEP 7 (TIA Portal) 中，如何使用 S7-1200 的“仅存储在装载存储器中”属性来组态数据块？
<https://support.industry.siemens.com/cs/ww/en/view/53034113>
使用配方功能保存 S7-1200 和 S7-1500 的重要数据
<https://support.industry.siemens.com/cs/ww/en/view/109479727>

3.6 符号寻址

3.6.1 用符号寻址替代绝对寻址

TIA Portal 针对符号编程进行了优化。具有诸多优点。通过符号寻址，用户不必注意内部数据存储就能编程。控制器处理数据的最佳存储位置。因此，用户可将精力完全集中于应用任务的解决方案上面。

优点

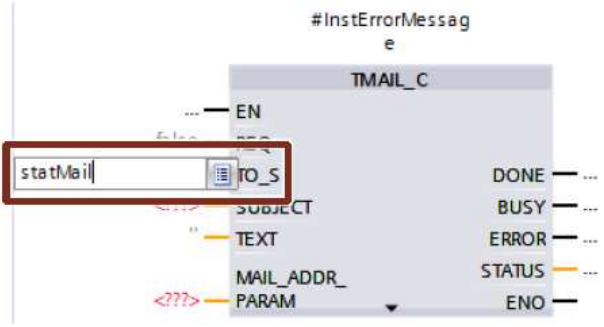
- 通过符号变量名称，程序更易于读取
- 在用户程序中的所有使用位置自动更新变量名称
- 无需手动管理程序数据的存储器管理（绝对寻址）
- 强大的数据访问功能
- 无需手动优化性能或程序大小
- 自动完成功能，快速输入符号
- 通过类型安全减少了程序错误（针对所有访问检查数据类型的有效性）

建议

- 不用担心数据存储问题
- 用符号的方式思维 输入每个函数、变量或数据的“描述性”名称，如 Pump_boiler_1、heater_room_4 等。这样，无需太多注释就可以方便地读取生成的程序。
- 为所有使用的变量指定直接符号名称，并随后通过右键单击对其进行定义。

示例

表 3-8：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何一个块。
2.	直接在指令输入处输入符号名称。 

步骤	说明
3.	<p>在该块旁边右键单击，然后在右键菜单中选择“定义变量...”(Define tag...)</p> 
4.	<p>定义变量。</p> 

如果要在一个程序段中定义多个变量，那么有一个节省时间的好方法。首先分配所有变量名称。然后使用步骤 4 的对话框同时定义所有变量。

注

更多信息，参见以下条目：
 在 STEP 7 (TIA Portal) 中针对 S7-1500 使用符号寻址有什么优势？
<https://support.industry.siemens.com/cs/ww/en/view/67598995>

3.6.2 ARRAY 数据类型和间接域访问

ARRAY 数据类型代表一种包含几个数据类型元素的数据结构。例如，ARRAY 数据类型适合存储配方、队列中的物料跟踪、循环过程数据采集、协议等。

图 3-28: 带有数据类型为 Integer (INT) 的 10 个元素的 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

通过变址寻址 (array ["index"]), 可以间接访问 ARRAY 中的具体元素。

图 3-29: 间接域访问



优点

- 通过 ARRAY 变址寻址，易于访问
- 无需创建复杂指针
- 可以快速创建和扩展
- 可在所有编程语言中使用

属性

- 结构化数据类型
- 数据结构由固定数目的数据类型相同的元素构成
- 也可以创建多维 ARRAY
- 通过在运行系统进行动态变址计算，可使用运行系统变量进行间接访问

建议

- 使用 **ARRAY** 代替指针（如 **ANY** 指针）来进行变址访问。这样会更容易读取程序，因为与存储器区域中的指针相比，具有符号名称的 **ARRAY** 更有意义。
- 作为运行变量，请使用 **DINT** 数据类型作为临时变量以获得最高性能。
- 使用“**MOVE_BLK**”指令将一个数组的各部分复制到另一个数组。
- 使用“**GET_ERR_ID**”指令捕捉数组内的访问错误。

注

更多信息，参见以下条目：

如何通过具有变址寻址功能的 S7-1500 实现数组访问？

<https://support.industry.siemens.com/cs/ww/en/view/67598676>

如何在 STEP 7 (TIA Portal) 中进行安全、间接寻址？

<https://support.industry.siemens.com/cs/ww/en/view/97552147>

在 STEP 7 (TIA Portal) 中，如何在数据类型为“Array of Bool”和“Word”的两个变量之间传送 S7-1500 的数据？

<https://support.industry.siemens.com/cs/ww/en/view/108999241>

3.6.3 形式参数 Array[*] (V14 或更高版本)

通过形式参数 **Array[*]**，可变长度的数组可以传送到函数和函数块。
使用指令“LOWER_BOUND”和“UPPER_BOUND”指令，可以确定数组限值。

优点

- 块可以处理不同长度的灵活数组
- 全符号编程，可读性最佳
- 不再需要对不同长度的数组进行指针编程

示例

图 3-30: 初始化不同的数组

The image shows a screenshot of the SIMATIC Manager software interface. At the top, the 'Main' block is visible with a variable declaration table:

Name	Data type
tempArray1	Array[0..125] of Real
tempArray2	Array[10..80] of Real

Below this, 'Network 1: Array initialization' is shown as a ladder logic network. It contains two 'InitArray' function blocks. The first block has its 'EN' input connected to '#tempArray1' and its 'quantityArray' output connected to the 'quantityArray' input of the second 'InitArray' block. The second block has its 'EN' input connected to '#tempArray2'. A large white arrow points from the 'quantityArray' input of the second 'InitArray' block to the 'InitArray' function block definition below.

The 'InitArray' function block definition is shown in a table below:

Name	Data type	Default value
quantityArray	Array[*] of Real	
tempLower	DInt	
tempUpper	DInt	
count	DInt	

At the bottom, the ladder logic network is expanded to show the following code:

```
1 #tempLower := LOWER_BOUND(ARR := #quantityArray, DIM := 1);
2 #tempUpper := UPPER_BOUND(ARR := #quantityArray, DIM := 1);
3
4 FOR #count := #tempLower TO #tempUpper DO
5   #quantityArray[#count] := 0.0;
6 END_FOR;
```

3.6.4 STRUCT 数据类型和 PLC 数据类型

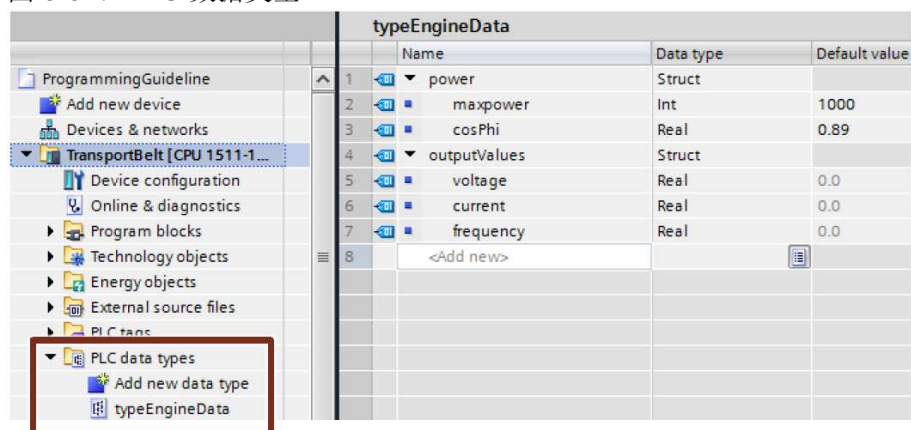
STRUCT 数据类型代表一种包含不同数据类型的元素的数据结构。在相应的块中进行结构声明。

图 3-31: 带有数据类型不同的元素的结构

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>		

与结构不同的是，PLC 数据类型在 TIA Portal 中是在整个控制器范围内定义的，可以集中更改。所有使用位置都会自动更新。使用之前，要在项目树中的“PLC 数据类型”(PLC data types) 文件夹中声明 PLC 数据类型。

图 3-32: PLC 数据类型



typeEngineData			
	Name	Data type	Default value
1	power	Struct	
2	maxpower	Int	1000
3	cosPhi	Real	0.89
4	outputValues	Struct	
5	voltage	Real	0.0
6	current	Real	0.0
7	frequency	Real	0.0
8	<Add new>		

优点

- PLC 数据类型的更改会在用户程序中的所有使用位置自动更新。
- 通过若干块之间的块接口进行简单的数据交换
- 在 PLC 数据类型中，可以声明具有指定长度的 **STRING** 变量（例如，String[20]）。从 TIA V14 开始，长度也可以使用全局常量（例如，String[LENGTH]）。
如果 **STRING** 变量没有定义长度，则最大长度为 255 个字符。

属性

- PLC 数据类型始终在字限制处终止（请见下面的图）。
- 请在以下情况下考虑此系统属性：
 - 在 I/O 区域中使用结构（参见 [3.6.5 通过 PLC 数据类型访问 I/O 区域](#)）。
 - 将 PLC 数据类型的帧用于通信。
 - 将 PLC 数据类型的参数记录用于 I/O 通信。
 - 使用未优化的块和绝对寻址。

图 3-33: PLC 数据类型始终在字限制处终止

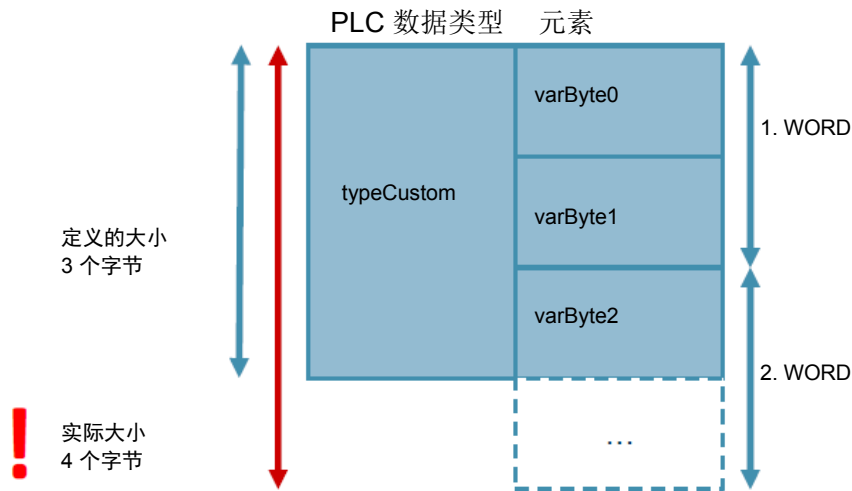
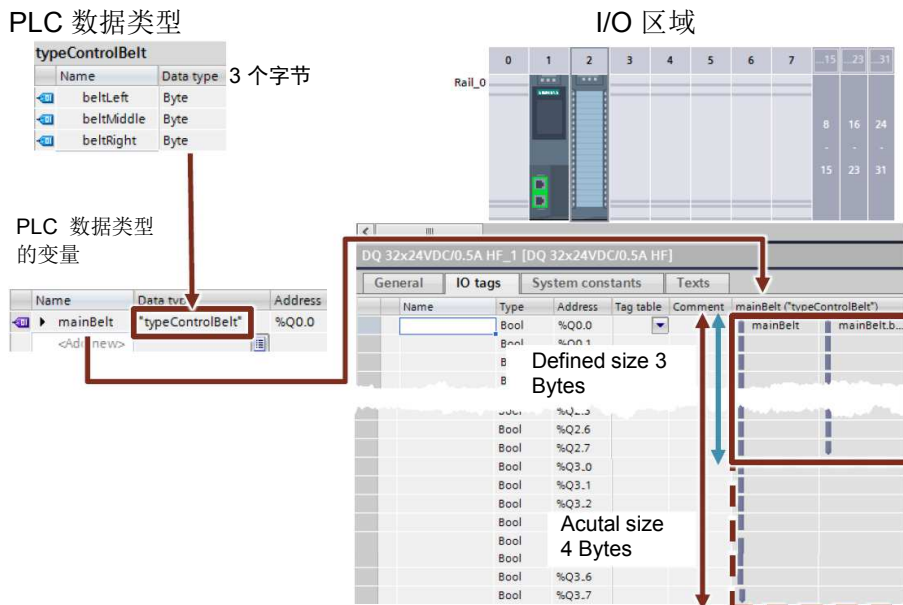


图 3-34: I/O 区域上的 PLC 数据类型



建议

- 使用 PLC 数据类型来总结多个关联的数据，如帧或电机数据（设定值、转速、旋转方向、温度等）

-
- 在用户程序中多次使用时，总是使用 PLC 数据类型（而不使用结构）。
 - 使用 PLC 数据类型将结构安排为数据块。
 - 使用 PLC 数据类型指定数据块的结构。PLC 数据类型可用于任意数目的数据块。您可以轻松、方便地创建尽可能多的相同结构的数据块，并在 PLC 数据类型上对它们集中调整。

注

更多信息，参见以下条目：

STEP 7 (TIA Portal) 和 S7-1200/S7-1500 的 PLC 数据类型 (LPD) 库

<https://support.industry.siemens.com/cs/ww/en/view/109482396>

如何在 STEP 7 (TIA Portal) 中将结构初始化到 S7-1500 的优化存储器区域中？

<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 Portal) 中，如何应用自己的数据类型 (UDT)？

<https://support.industry.siemens.com/cs/ww/en/view/67582844>

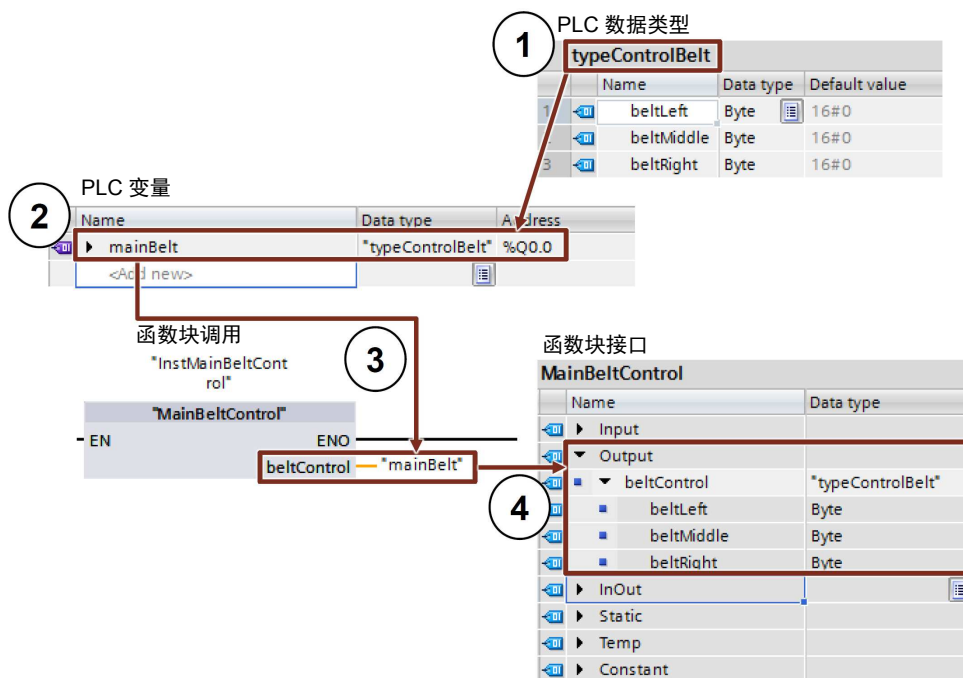
为什么在调用块时应传输 S7-1500 的整个结构而不是许多单个部分？

<https://support.industry.siemens.com/cs/ww/en/view/67585079>

3.6.5 通过 PLC 数据类型访问 I/O 区域

通过 S7-1500 控制器，可以创建 PLC 数据类型并将它们用于对输入和输出的结构化及符号访问。

图 3-35: 通过 PLC 数据类型访问 I/O 区域



7. 带有全部所需数据的 PLC 数据类型
8. 具有所创建的 PLC 数据类型以及 I/O 数据区域的起始地址（%Ix.0 或 %Qx.0，如 %I0.0、%Q12.0 等）的 PLC 变量
9. 将 PLC 变量作为实际参数传输至函数块
10. 函数块的输出类型是所创建的 PLC 数据类型

优点

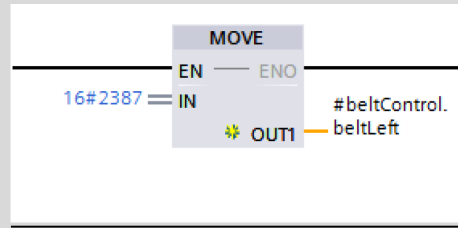
- 编程效率高
- 由于使用 PLC 数据类型，便于多次重复使用

建议

- 使用 PLC 数据类型来访问 I/O 区域，例如，通过符号方式接收和发送变频器报文。

注

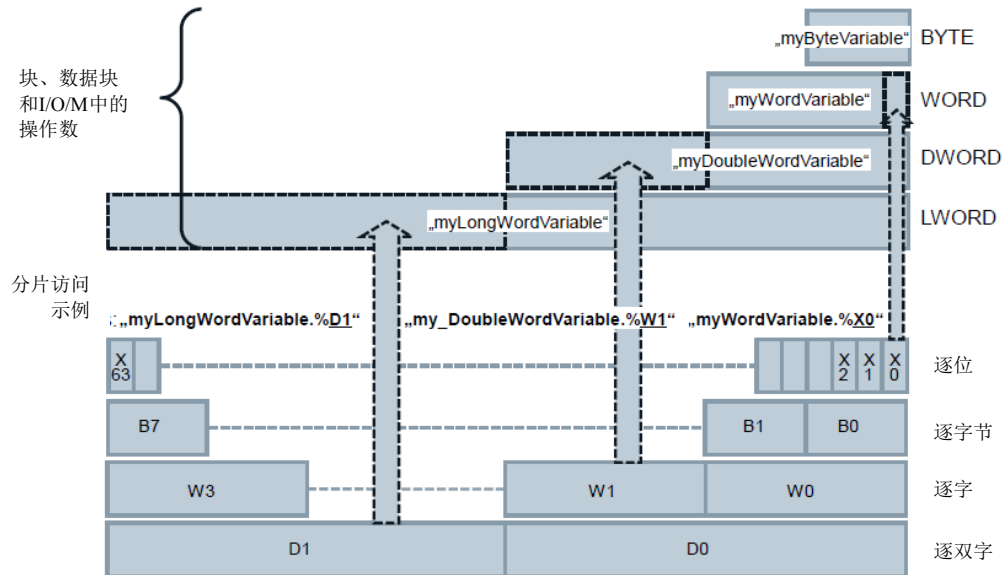
也可以在用户程序中直接访问变量的 PLC 数据类型的具体元素：



3.6.6 分片访问

对于 S7-1200/1500 控制器，您可以访问数据类型为 Byte、Word、DWord 或 LWord 的变量的存储器区域。将存储器区域（如字节或字）划分为更小的存储器区域（如布尔）也称为分片。下图显示了对操作数执行的符号位、字节和字访问。

图 3-36: 符号位、字节、字、双字分片访问



优点

- 编程效率高
- 无需在变量声明中进行额外定义
- 易于访问（例如，控制位）

建议

- 使用分片访问而不是通过访问操作数中的特定数据区域进行 AT 构造。

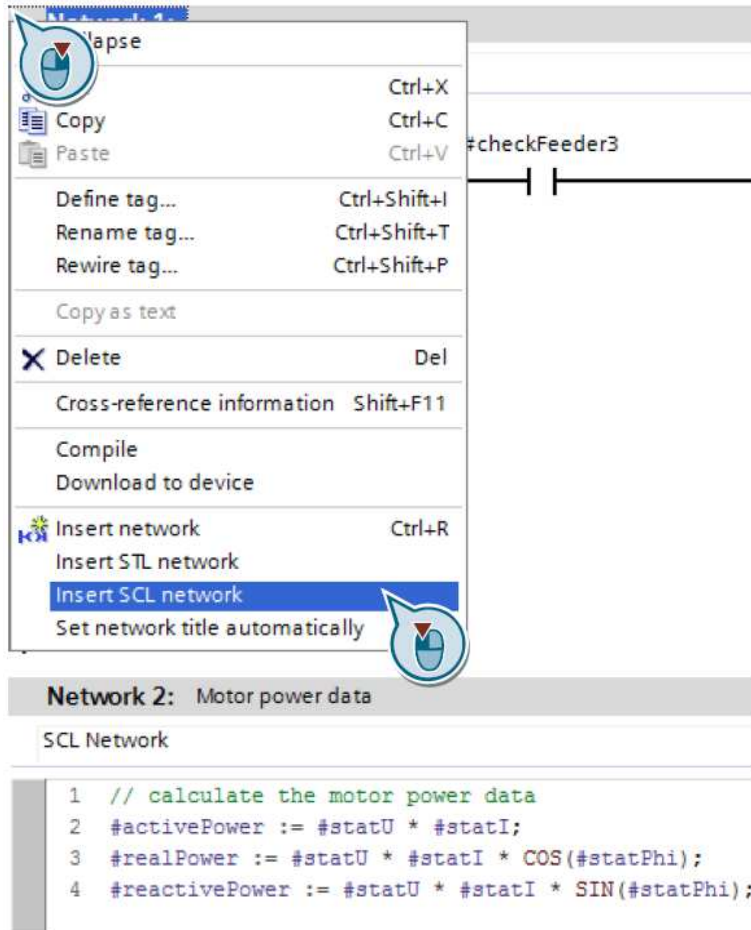
注

更多信息，参见以下条目：
在 STEP 7 (TIA Portal) 中，如何逐位、逐字节、逐字和以符号方式访问未结构化的数据类型？
<https://support.industry.siemens.com/cs/ww/en/view/57374718>

3.6.7 LAD 和 FBD 中的 SCL 程序段（V14 及更高版本）

借助 SCL 程序段，可以在 LAD 和 FBD 中进行计算，而对于 LAD 和 FBD 指令，只能相当费力地进行编程。

图 3-37：插入 SCL 程序段



优点

- 通过高效编程节省时间
- 符号编程，代码清晰

属性

- 支持所有 SCL 指令
- 支持注释

建议

- 使用 SCL 程序段而非指令（如 ADD、SUBB 等）在 LAD 和 FBD 进行算术运算。

3.7 库

通过 TIA Portal，可以从可方便地重新使用的不同项目元素创建独立的库。

优点

- 简单存储在 TIA Portal 中组态的数据：
 - 完整设备（控制器、人机界面、驱动器等）
 - 块、变量表、PCL 数据类型、监控表等
 - HMI 画面、HMI 变量、脚本等
- 通过库进行跨项目的数据交换
- 集中更新库元素
- 库元素版本控制
- 通过系统支持的依赖性，错误来源减少

建议

- 创建主副本以方便地重复使用块、硬件组态、HMI 画面等
- 创建用于在系统支持下重复使用库元素的类型：
 - 块的版本控制
 - 集中更新所有使用位置
- 使用全局库以便与其它用户交换数据并作为多个用户同时使用时的中央存储位置。
- 组态全局库的存储位置，以便它能够在 TIA Portal 启动时自动打开。

有关详细信息，请访问网址：

<https://support.industry.siemens.com/cs/ww/en/view/100451450>

注

更多信息，参见以下条目：

可以将 STEP 7 (TIA Portal) 和 WinCC (TIA Portal) 的哪些元素作为类型或主副本存储在库中？

<https://support.industry.siemens.com/cs/ww/en/view/109476862>

在 STEP 7 (TIA Portal) 中，如何使用写访问权限打开全局库？

<https://support.industry.siemens.com/cs/ww/en/view/37364723>

3.7.1 库类型和库元素

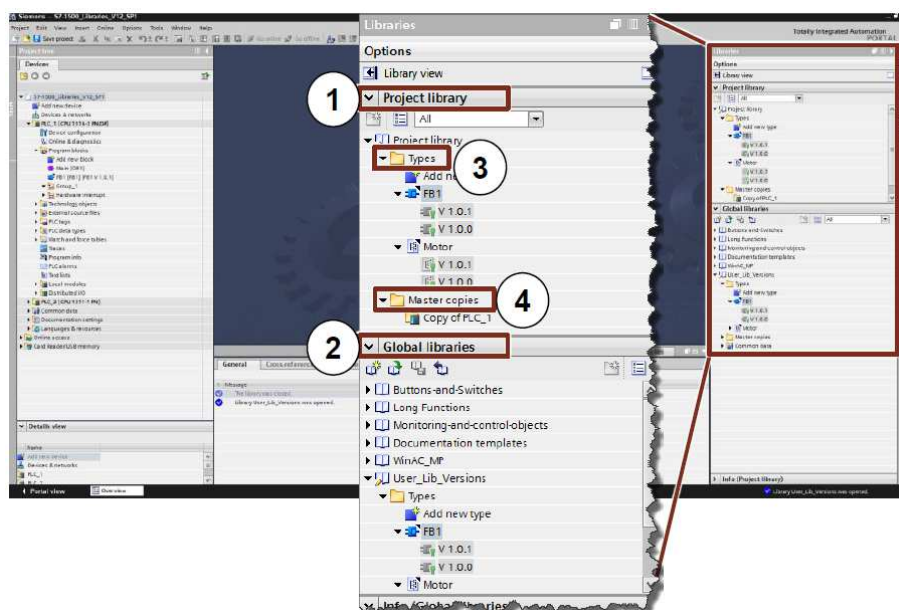
通常有两种不同类型的库：

- 项目库
- 全局库

每种库包含两个存储类型：

- 类型
- 主副本

图 3-38: TIA Portal 中的库



(1) 项目库

- 集成在项目中并通过项目来管理
- 允许在项目内重复使用

(2) 全局库

- 独立库
- 可在多个项目内使用

一个库包含两种不同类型的库元素存储：

(3) 主副本

- 库中组态元素（如块、硬件、PLC 变量表等）的副本
- 副本不与项目中的元素相连。
- 主副本也可由多个组态元素组成。

(4) 类型

- 类型与项目中的使用位置相连。更改类型后，项目中的使用位置可自动更新。

- 支持的类型是控制块（函数、函数块）、PLC 数据类型、HMI 画面、HMI 面板、HMI UDT、脚本）。
- 附属元素将自动类型化。
- 类型具有版本控制：可通过创建较新版本来更改。
- 控制器内使用的类型只能有一个版本。

3.7.2 类型概念

通过类型概念，可以创建可在多个工厂设备中使用的标准化自动化功能。类型概念在版本控制和更新功能方面为用户提供支持。

可以在用户程序中使用库中的类型。优点如下：

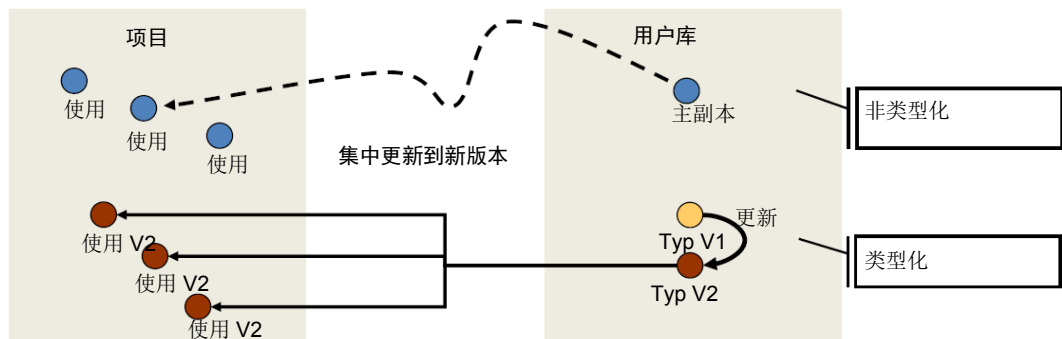
优点

- 集中更新项目中的所有使用位置
- 无法对类型的使用位置进行不需要的修改。
- 系统通过阻止不需要的删除操作来保证类型保持一致。
- 若删除了某个类型，则用户程序中的所有使用位置都被删除。

属性

通过使用类型，可以集中进行更改，并对整个项目中的类型进行更新。

图 3-39: 使用用户库实现类型化



- 总是对类型进行标记，以便更好识别

3.7.3 CPU 和 HMI 的可类型化对象的差异

控制器和 HMI 的可类型化对象之间与系统相关的差异：

表 3-9：控制器和 HMI 的类型差异

控制器	HMI
将附属控制元素类型化。	不将附属 HMI 元素类型化。
将附属控制元素实例化。	不将附属 HMI 元素实例化。
在测试环境中编辑控制元素。	在测试环境中编辑 HMI 画面和 HMI 脚本。直接在库中编辑面板和 HMI – UDT，无需测试环境。

下面的示例中提供了有关库的操作的信息。

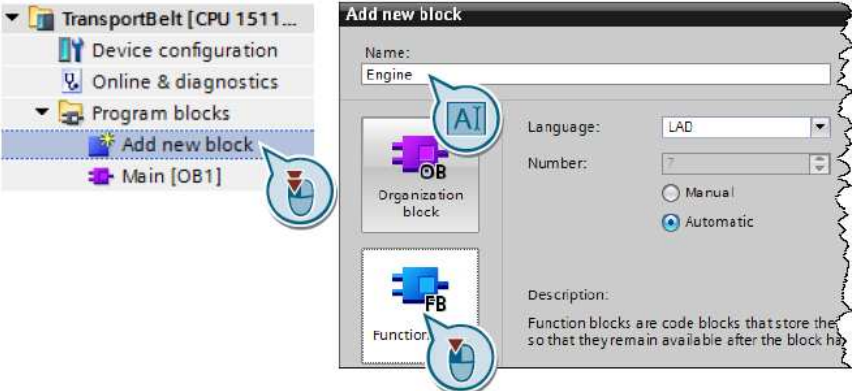
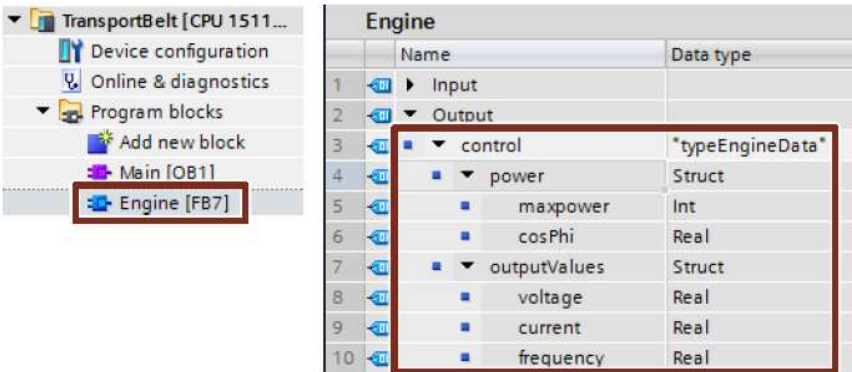
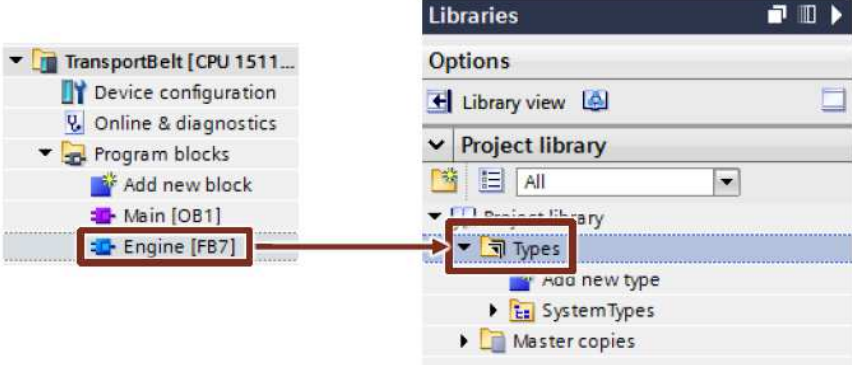
3.7.4 块的版本控制

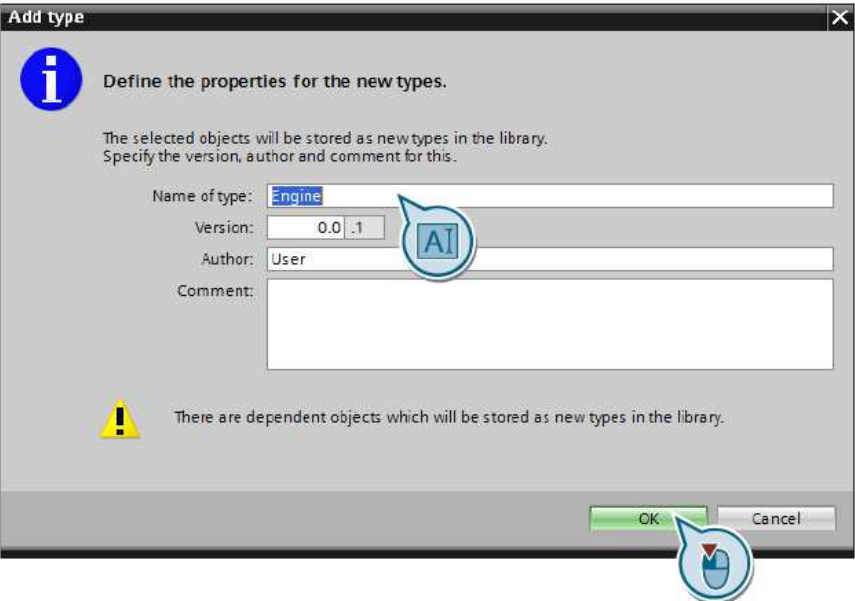
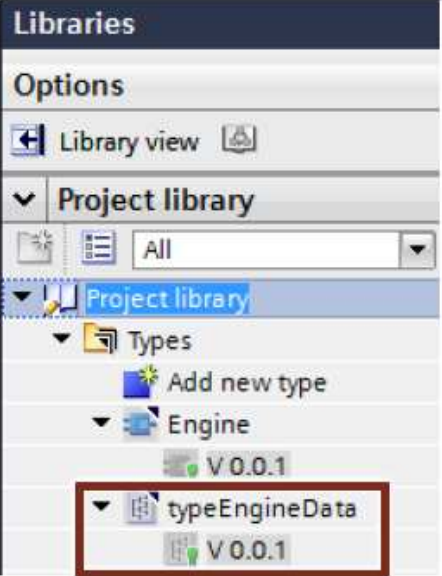
示例：创建类型

下面的示例说明了如何使用库的基本函数。

表 3-10：创建类型

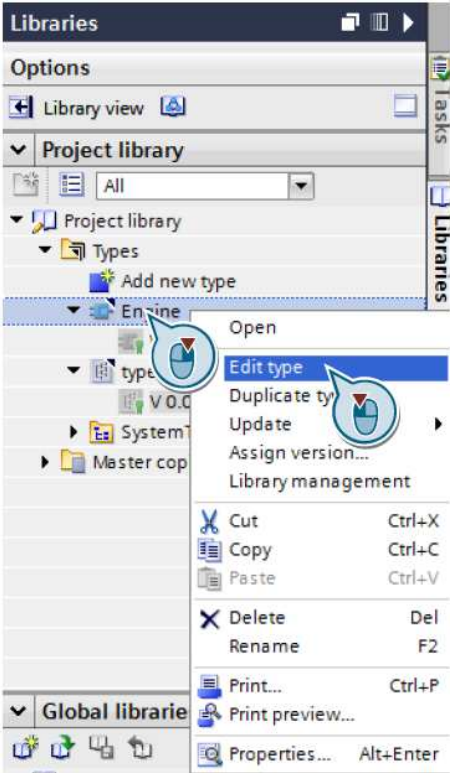
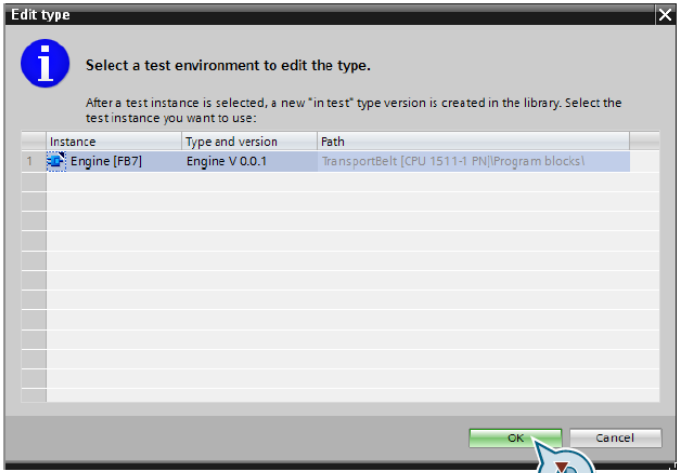
步骤	说明																																				
1.	<p>使用“添加新数据类型”(Add new data type) 创建新 PLC 数据类型并创建某些变量。该类型随后为附属类型。</p> <table border="1" data-bbox="734 1187 1324 1433"> <thead> <tr> <th colspan="4">typeEngineData</th> </tr> <tr> <th>Name</th> <th>Data type</th> <th colspan="2">Default value</th> </tr> </thead> <tbody> <tr> <td>power</td> <td>Struct</td> <td colspan="2"></td> </tr> <tr> <td>maxpower</td> <td>Int</td> <td colspan="2">1000</td> </tr> <tr> <td>cosPhi</td> <td>Real</td> <td colspan="2">0.89</td> </tr> <tr> <td>outputValues</td> <td>Struct</td> <td colspan="2"></td> </tr> <tr> <td>voltage</td> <td>Real</td> <td colspan="2">0.0</td> </tr> <tr> <td>current</td> <td>Real</td> <td colspan="2">0.0</td> </tr> <tr> <td>frequency</td> <td>Real</td> <td colspan="2">0.0</td> </tr> </tbody> </table>	typeEngineData				Name	Data type	Default value		power	Struct			maxpower	Int	1000		cosPhi	Real	0.89		outputValues	Struct			voltage	Real	0.0		current	Real	0.0		frequency	Real	0.0	
typeEngineData																																					
Name	Data type	Default value																																			
power	Struct																																				
maxpower	Int	1000																																			
cosPhi	Real	0.89																																			
outputValues	Struct																																				
voltage	Real	0.0																																			
current	Real	0.0																																			
frequency	Real	0.0																																			

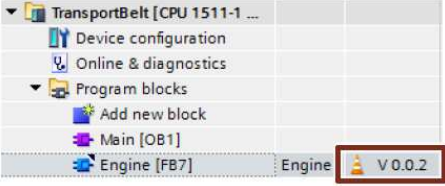
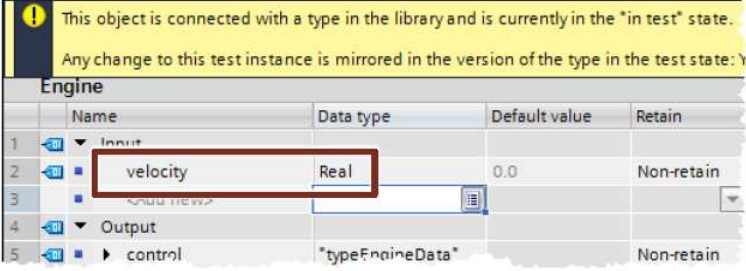
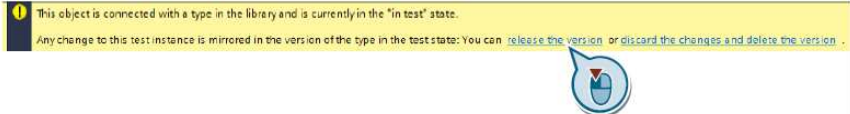
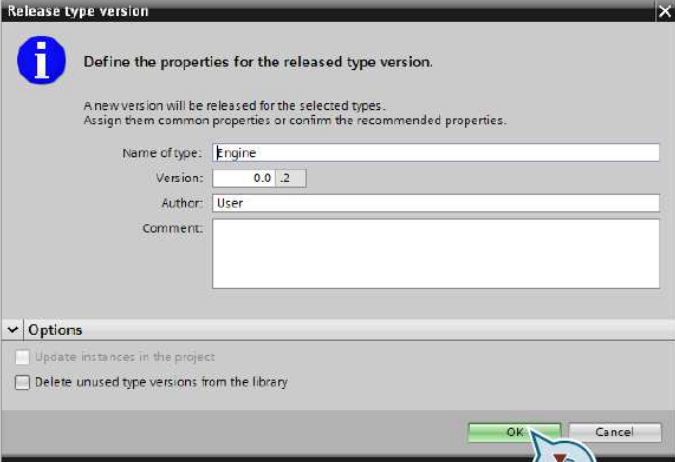
步骤	说明
2.	<p>使用“添加新块”(Add new Block) 创建新函数块。这是上层类型。</p> 
3.	<p>定义已创建的数据类型的输出变量。因此，PLC 数据类型附属于该函数块。</p> 
4.	<p>将该函数块拖放到项目库中的“类型”(Types) 文件夹中。</p> 

步骤	说明
5.	<p>可以分配：类型名称、版本、作者和注释，然后按“确定”(OK) 确认。</p> 
6.	<p>附属 PLC 数据类型也自动存储在库中。</p> 

示例：更改类型

表 3-11：更改类型

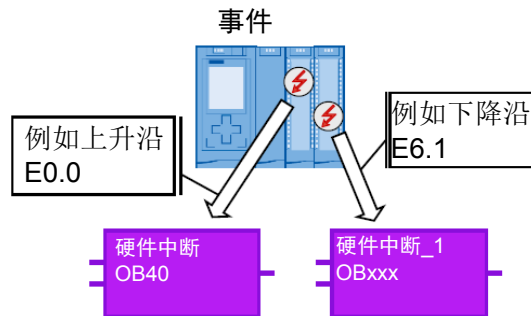
步骤	说明
1.	<p>右键单击“项目库”(Project library) 中的块，然后选择“编辑类型”(Edit type)。</p> 
2.	<p>选择要用作测试环境的控制器，然后按“确定”(OK) 确认该对话框。</p>  <p>如果项目中的多个控制器使用选定的块，则必须选择一个控制作为测试环境。</p>

步骤	说明
3.	块被打开。新版本的块已创建。 
4.	添加一个输入变量。 在此位置，可通过将项目加载到控制器上来测试对块的更改。完成块的测试后，继续执行以下步骤。 
5.	单击“发布版本”(Release the version) 按钮。 
6.	随即会打开一个对话框。可在其中存储与版本相关的注释。按“确定”(OK) 确认该对话框。  <p>如果该块在项目的不同控制器中有多个使用位置，则可以同时更新这些使用位置：“更新项目中的实例”(Update instances in the project)。如果不再需要该元素的就版本，则可通过单击“从库中删除未使用的类型版本”(Delete unused type versions from library) 将它们删除。</p>

3.8 提高了硬件中断性能

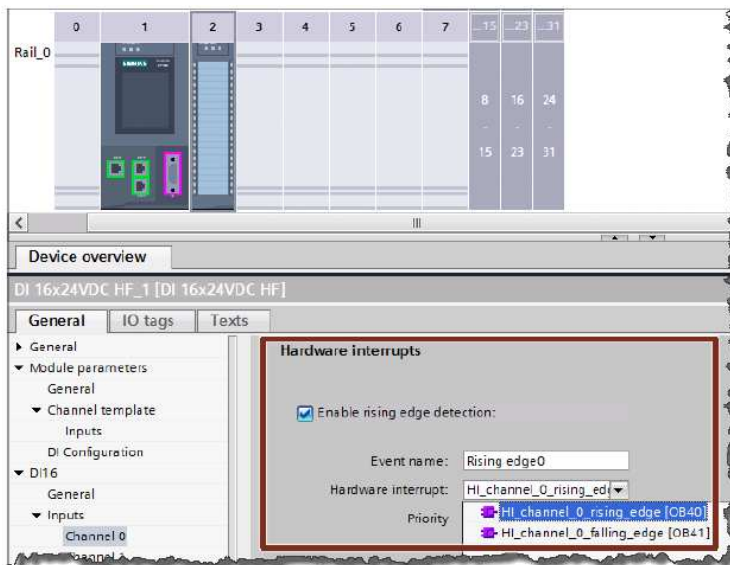
硬件中断等事件会影响用户程序的处理。当需要控制器对硬件事件做出快速响应时（例如，数字量输入模块的上升沿），请组态硬件中断。对于每个硬件中断，可以编程单独的 OB。发生硬件中断时，此 OB 由控制器的操作系统调用。因此，控制器循环被中断并在处理硬件中断之后继续。

图 3-40：硬件中断调用 OB



下图显示了一个数字量输入模块的硬件组态中的“硬件中断”。

图 3-41：组态硬件中断



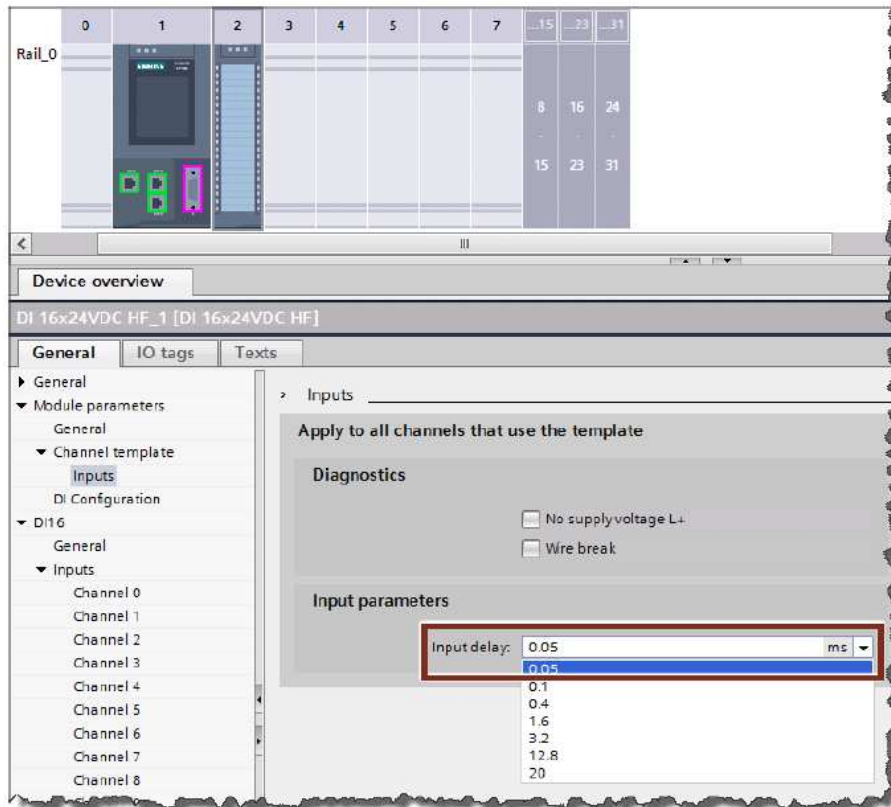
优点

- 系统对事件（上升沿、下降沿等）快速响应
- 每个事件都可启动单独的 OB。

建议

- 使用硬件中断来编程对硬件事件的快速响应。
- 如果编程了硬件中断后系统响应仍不够快，那么仍可以加快响应。在模块中设置尽可能小的输入延迟。对事件的响应总是仅当输入延迟过后才能发生。输入延迟用于过滤输入信号，例如，用于补偿触点弹跳或颤动等故障。

图 3-42: 设置输入延迟



3.9 其它性能建议

下面是一些能够让控制器的程序处理更加快速的一般建议。

建议

请注意以下对 S7-1200/1500 控制器进行编程以取得高性能的建议：

- LAD/FBD：禁用块的“检测 ENO”(evaluate ENO)。这样就可避免在运行时进行测试。
- STL：请勿使用寄存器，因为 S7-1500 仅出于兼容性原因对地址和数据寄存器进行仿真。

注

更多信息，参见以下条目：

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 Portal) 以及 S7- 1200/S7-1500 CPU 中的性能？

<https://support.industry.siemens.com/cs/ww/en/view/37571372>

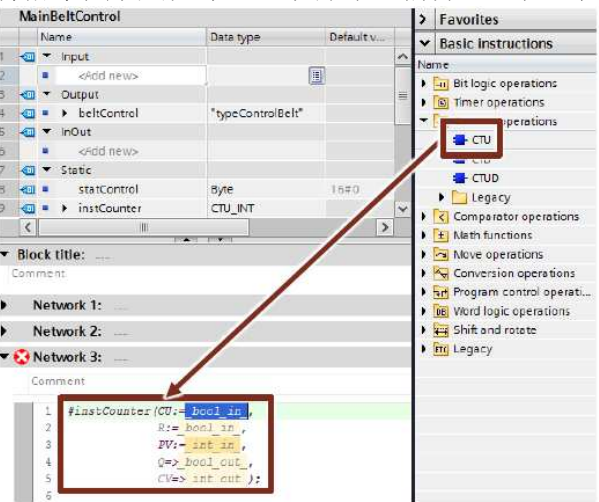
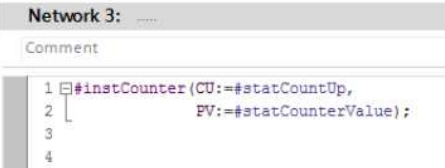
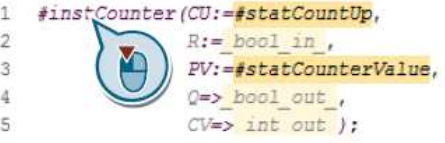
3.10 SCL 编程语言：提示与技巧

3.10.1 使用调用模板

编程语言的很多指令都提供了包含现有形式参数列表的模板。

示例

表 3-12: 方便地扩展调用模板

步骤	说明
1.	<p>将指令从库拖放到 SCL 程序中。编辑器显示整个调用模板。</p> 
2.	<p>填写所需的参数“CU”和“PV”，然后按“Return”按钮结束输入。</p>
3.	<p>编辑器自动精简调用模板。</p> 
4.	<p>如果您希望稍后再次编辑完整该调用模板，请按以下步骤操作。在调用模板中的任何位置单击，然后按“Ctrl+Shift+空格”键。此时已计入调用模板模式。编辑器再次展开调用模板。可使用“Tab”按钮在参数中导航。</p> 
5.	<p>注：在“调用模板”(Call Template) 模式下，写入内容显示为斜体。</p>

3.10.2 什么指令参数是强制性参数？

如果您展开调用模板，那么将通过颜色代码来简单明了地显示指令的哪些形式参数是可选参数，哪些不是可选参数。
强制性参数显示为深色。

3.10.3 整个变量名称的拖放

在 SCL 编辑器中，也可以使用拖放功能。另外还支持变量名称的拖放。如果要将一个变量替换为另一个变量，请按以下步骤操作。

表 3-13: SCL 中变量的拖放

步骤	说明
1.	<p>将变量拖放到程序中要替换的变量。保持住该变量 1 秒以上，然后将其释放。</p>  <p>整个变量被替换。</p>

3.10.4 使用关键字 REGION (V14 或更高版本) 结构化

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

优点

- 更为直观
- 即使在较大块中也很容易导航
- 现成的代码片段可以折叠。

属性

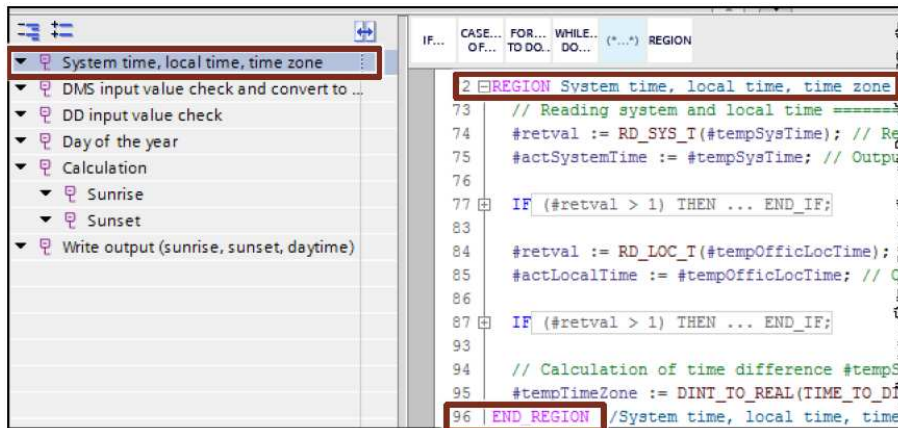
REGION 可以嵌套。

建议

使用关键字 REGION 来结构化 SCL 块。

示例

图 3-43: SCL 编辑器



3.10.5 正确使用 FOR、REPEAT 和 WHILE 循环

循环的使用分不同的版本和应用。以下示例显示了其中的差异。

属性：FOR 循环

FOR 循环按照定义的运行次数运行。循环变量在开始时被赋予一个起始值。之后，循环变量在每次循环运行中以指定的步长递增至结束值。

出于性能原因，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。

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

通过 **EXIT** 指令，可以随时中断循环。

属性：WHILE 循环

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，如果条件没有立即满足，则不执行循环。每个变量都可以针对循环代码中的下一次运行进行调整。

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

属性：REPEAT 循环

REPEAT 循环由终止条件结束。在循环代码末尾检查终止条件。这意味着，循环至少运行一次。每个变量都可以针对循环代码中的下一次运行进行调整。

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

建议

- 如果循环变量明确定义，则使用 **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 有效使用 CASE 指令

通过 SCL 中的 CASE 指令，可精确跳转到选定的 CASE 块条件。执行该 CASE 块后，指令执行完毕。例如，这样就可以更具体和更方便地检查经常需要使用的值范围。

示例

```
CASE #myVar OF
  5:
      #Engine(#myParam);
  10,12:
      #Transport(#myParam);
  15:
      #Lift(#myParam);
  0..20:
      #Global(#myParam);

// 对于值 5、10、12 或 15，从不会调用全局变量！
ELSE
END_CASE;
```

注 CASE 指令也可与 CHAR、STRING 数据类型以及元素配合使用（参见 [2.8.5 数据类型 VARIANT](#) 中的示例）。

3.10.7 不操作 FOR 循环的循环计数器

SCL 中的 FOR 循环是纯计数器循环，即进入该循环后，迭代次数是固定的。在 FOR 循环中，无法更改循环计数器。
通过 EXIT 指令，可以随时中断循环。

优点

- 编译器可更好地优化程序，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 反向 FOR 循环

在 SCL 中，也可以反向或以其它步长来递增 FOR 循环的索引。为此，请在循环入口处使用可选的“BY”关键字。

示例

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

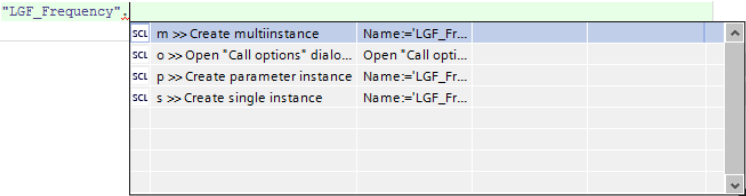
如示例中所示，如果将“BY”定义为“-2”，则每次迭代时，计数器递减 2。如果省略“BY”，则“BY”的默认设置为 1。

3.10.9 方便地创建实例进行调用

如果希望使用键盘，则可以简单地在 SCL 中创建实例。

示例

表 3-14: 方便地创建实例

步骤	说明
1.	指定块名称，后面跟一个句点 (.)。自动编译后，将显示以下结果。 
2.	顶部显示了现有实例。另外，也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

在 SCL 中，可以像处理普通数字那样计算时间变量，也就是说，无需寻找其它函数（如 T_COMBINE），但是可以使用简单运算。此方法叫做“操作数重载”。SCL 编译器自动使用合适的函数。您可以针对相应时间类型使用一种合理运算，因此可更高效地编程。

示例

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

下表总结了重载运算符及其使用该运算符的运算：

表 3-15: SCL 的重载操作数

重载操作数	运算
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

程序员经常想到使用 IF-THEN-ELSE 指令。但这常会导致程序中不必要的结构。

示例

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

建议

请记住，对于布尔型请求，直接赋值通常更有效。整个结构可以用一行来编程。

示例

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

4 编程不依赖于硬件

要确保块不用进一步调整就可所有控制器上使用，务必不要使用依赖于硬件的函数和属性。

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

- 仅使用要运行该程序的控制器所支持的数据类型。

表 4-1: 符合标准 EN 61131-3 的基本数据类型

	说明	S7-300/400	S7-1200	S7-1500
位数据类型	<ul style="list-style-type: none">• BOOL• BYTE• WORD• DWORD	√	√	√
	<ul style="list-style-type: none">• LWORD	-	-	√
字符类型	<ul style="list-style-type: none">• CHAR (8 位)	√	√	√
数值数据类型	<ul style="list-style-type: none">• INT (16 位)• DINT (32 位)• REAL (32 位)	√	√	√
	<ul style="list-style-type: none">• SINT (8 位)• USINT (8 位)• UINT (16 位)• UDINT (32 位)• LREAL (64 位)	-	√	√
	<ul style="list-style-type: none">• LINT (64 位)• ULINT (64 位)	-	-	√
时间类型	<ul style="list-style-type: none">• TIME• DATE• TIME_OF_DAY	√	√	√
	<ul style="list-style-type: none">• S5TIME	√	-	√
	<ul style="list-style-type: none">• LTIME• L_TIME_OF_DAY	-	-	√

表 4-2 由其它数据类型组成的数据组

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

表 4-3: 在块之间传送的形式参数的参数类型

	说明	S7-300/400	S7-1200	S7-1500
指针	• POINTER • ANY	√	no	√ 1)
	• VARIANT	-	√	√
块	• TIMER • COUNTER	√	√ 2)	√
	• BLOCK_FB • BLOCK_FC	√	-	√
	• BLOCK_DB • BLOCK_SDB	√	-	-
	• VOID	√	√	√
PLC 数据类型	• PLC DATA TYPE	√	√	√

- 1) 对于优化的访问，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

- 与出于兼容性原因而未优化的位存储器区域相比，优化的全局数据块明显功能更强大。

建议

- 位存储器（也包括系统和时钟标志）的处理会有问题，因为每个控制器都具有大小不同的标志区域。请勿将位存储器用于编程，而总是要使用全局数据块。这样，程序始终能够通用。

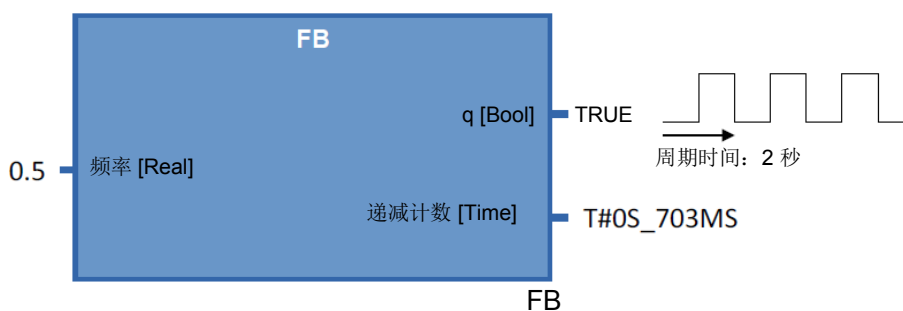
4.3 “时钟位”编程

建议

为了编程时钟存储器，硬件组态必须始终正确。
使用编程的块作为时钟发生器。下面是使用 SCL 编程语言编程时钟发生器的示例。

示例

编程块具有以下功能。预设所需的频率。“Q”输出是一个布尔值，可在所需频率范围内切换。“Countdown”输出用于输出“Q”的当前状态的剩余时间。
如果所需频率小于或等于 0.0，则输出 Q = FALSE，Countdown = 0.0。



注

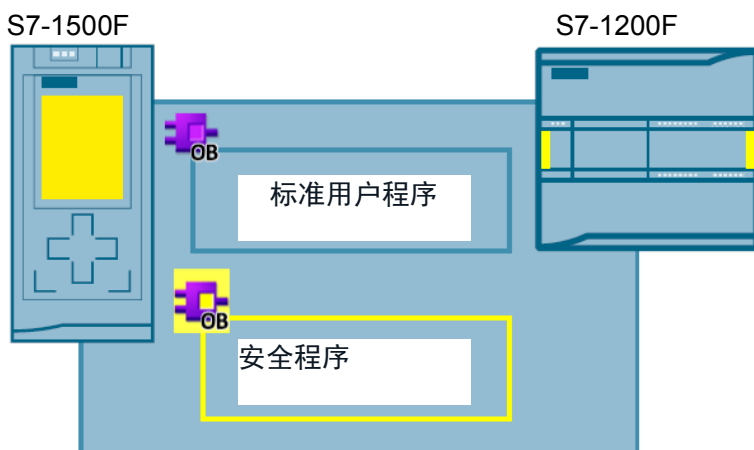
全部编程示例，参见以下条目：
<https://support.industry.siemens.com/cs/ww/en/view/109479728>

5 TIA Portal 中的 STEP 7

5.1 简介

TIA Portal V13 SP1 或更高版本由故障安全型 S7-1200F 和 S7-1500F CPU 支持。在这些控制器中，在一台设备中可以进行标准以及故障安全型编程。编程故障安全用户程序时，将使用 SIMATIC STEP 7 Safety (TIA Portal) 可选软件包。

图 5-1: 标准和程序



优点

- 通过一种组态工具，可对标准和故障安全应用统一编程：TIA Portal
- 编程与使用 LAD 和 FBD 的编程类似
- 统一的诊断和在线功能

注 故障安全并不意味着程序没有错误。程序员负责采用正确的编程逻辑。故障安全的意思就是确保在控制器中正确处理故障安全用户程序。

注 有关安全主题的其他信息（如安全程序的安全要求和原则），请见以下文章：

TIA Portal – 最重要的文档与链接概览 – 安全性

<https://support.industry.siemens.com/cs/ww/en/view/90939626>

应用与工具 – Safety Integrated

<https://support.industry.siemens.com/cs/ww/en/ps/14675/ae>

STEP 7 Safety (TIA Portal) – 手册

<https://support.industry.siemens.com/cs/ww/en/ps/14675/man>

5.2 术语

本文档一致使用具有以下含义的术语。

表 5-1: 安全术语

术语	说明
标准用户程序	标准用户程序是与 F 编程相关的程序部分。
安全程序 (F 程序、故障安全用户程序)	故障安全用户程序是独立于控制器进行处理的程序部分。 为了与标准用户程序的块和指令区分, 故障安全程序块和指令在软件用户界面上都标为黄色 (例如, 在项目树中)。 F-CPU 和 F-I/O 的故障安全参数在硬件组态中标为黄色。

5.3

安全程序的组件

安全程序总是包含用户生成的 F 块和系统生成的 F 块以及“安全管理”编辑器。

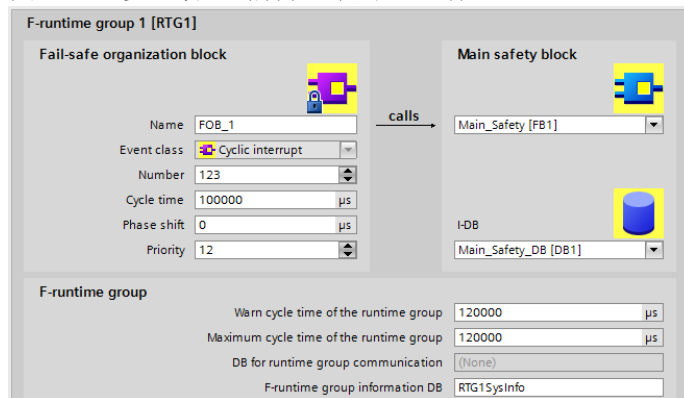
表 5-2: 安全程序的组件

说明	画面
<p>1. “安全管理”编辑器</p> <ul style="list-style-type: none"> - 安全程序的状态 - F 集合签名 - 安全运行状态 - 创建/组织 F 运行组 - 有关 F 块的信息 - 有关符合 F 标准的 PLC 数据类型的信息 - 定义/更改访问保护 	
<p>2. 用户创建的 F 块</p>	
<p>3. 系统生成的 F 运行块</p> <ul style="list-style-type: none"> - 块包含有关 F 运行组的状态信息。 	
<p>4. 系统生成的 F-I/O 数据块</p> <ul style="list-style-type: none"> - 块包含用于检测 F 模块的变量。 	
<p>5. “编译器块”</p> <p>系统生成的验证块</p> <ul style="list-style-type: none"> - 这些块在控制器的后台运行，负责安全处理安全程序。 - 用户不能处理这些块。 	

5.4 F 运行组

始终在具有定义的周期的 F 运行组中处理安全程序。F 运行组包含用于调用“主安全块”(Main safety block) 的“故障安全组织块”(Fail-safe organization block)。所有用户生成的安全函数都从“主安全块”调用。

图 5-2: “安全管理”编辑器中的 F 运行组



优点

- 在“安全管理器”中，可以方便地创建和组态运行组。
- 运行组中的 F 块是自动创建的。

属性

- 可以创建最多两个 F 运行组。

5.5 F 签名

每个 F 组件（站、I/O、块）都具有唯一 F 签名。使用 F 签名，可迅速检测到 F 设备组态、F 块或整个站是否仍符合原始组态或编程。

优点

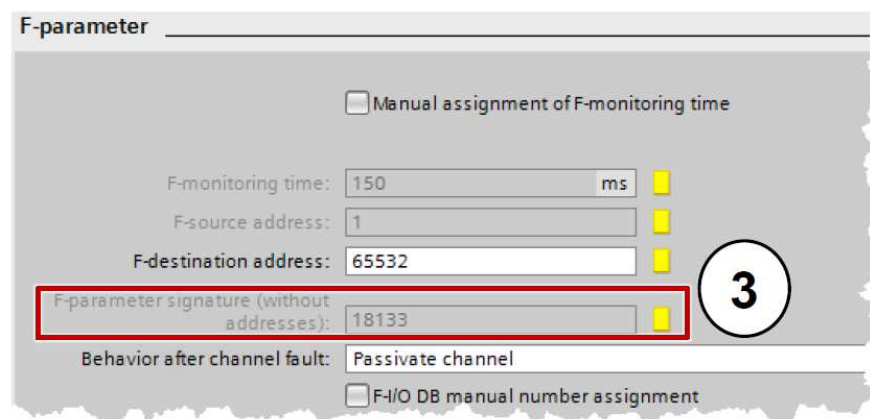
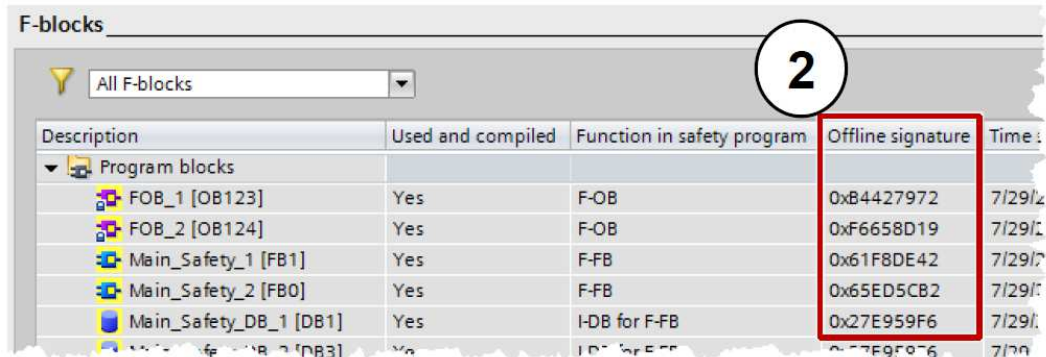
- 方便而快速地比较 F 块和 F 设备组态

属性

- F 参数签名（不带 F-I/O 地址）
 - 只能通过调整参数来更改。
 - 更改 PROFIsafe 地址时保持不变。不过，站的 F 集合签名会改变。
- 仅当 F 块中的逻辑发生变化时，才会更改 F 块签名。
- 更改以下内容时，F 块签名保持不变：
 - 块编号
 - 块接口
 - 块版本

示例

图 5-3: F 签名示例



1. “安全管理”编辑器中站的 F 集合签名
2. “安全管理”编辑器中的 F 块签名（也可从块的属性读取）
3. “设备与网络”(Devices & Networks) 中“设备视图”(Device view) 中的 F 参数签名

注 对于 S7-1500F 控制器，可以直接在安装的显示屏上或在集成的 Web 浏览器中读取 F 总体签名。

5.6 分配 F-I/O 的 PROFIsafe 地址

每个 F-I/O 设备都有一个用于标识以及与 F 控制器通信的 PROFIsafe 地址。在分配 PROFIsafe 地址时，可以有两种不同组态。

表 5-3: 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 DIL 开关，直接在模块上分配 PROFIsafe 地址。 TIA Portal 的设备组态中以及外围设备的 DIL 开关位置处的 PROFIsafe 地址必须相同。	仅通过 TIA Portal 分配 PROFIsafe 地址。 组态的 PROFIsafe 地址将加载到模块的智能编码模块上。

优点

- 无需在 ET 200MP 和 ET 200SP 上重新分配 PROFIsafe 地址，就可更换 F 模块。模块更换过程中，智能编码模块保留在基本单元中。
- 由于 TIA Portal 可显示 PROFIsafe 地址错误分配警告，可实现简便组态。
- 在 ET 200SP 中，可同时分配所有 F 模块的 PROFIsafe 地址。

注

有关分配 F-I/O 的 PROFIsafe 地址的详细信息，请见以下文章：
SIMATIC 工业软件 SIMATIC Safety – 组态和编程
<https://support.industry.siemens.com/cs/ww/en/view/54110126>

5.7 F-I/O 的检测

相应 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，可对状态进行检测和处理。S7-1200F/1500F 和 S7-300F/400F 之间存在以下差别。

表 5-4: S7-300F/400F 和 S7-1500F 的 F-I/O DB 中的变量

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

* QBAD_I_xx 和 QBAD_O_xx 显示通道值的有效性，符合 S7-1200F/1500F 处的反相值（下面一章提供了详细信息）。

5.8 值状态 (S7-1200F/1500F)

除诊断信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的有效性（值状态）的信息。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：值状态显示相应通道值的有效性。

- 1: 输出该通道的有效过程值。
- 0: 输出该通道的替代值。

表 5-5: Q_BAD (S7-300F/400F) 和值状态 (S7-1200F/1500F) 之间的差别

情形	QBAD (S7-300F/400F)	值状态 (S7-1200F/1500F)
F-I/O 处的有效值（无错误）	FALSE	TRUE
发生通道错误	TRUE	FALSE
通道错误消失 (ACK_REQ)	TRUE	FALSE
故障确认 (ACK_REI)	FALSE	TRUE

属性

- 值状态输入到输入与输出的过程映像中。
- 只能从相同的 F 运行组来访问 F-I/O 的通道值和值状态。

建议

- 为提高可读性，可在末尾分配“_VS”（如“TagIn1VS”）以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 5-6: 以 F-DI 8x24VDC HF 为例，过程映像中的值状态位

F- CPU 中的字节	F- CPU 中分配的位							
	7	6	5	4	3	2	1	0
x + 0	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0
x + 1	D17 的值状态	D16 的值状态	D15 的值状态	D14 的值状态	D13 的值状态	D12 的值状态	D11 的值状态	D10 的值状态

x = 模块起始地址

注 有关所有 ET 200SP 模块的值状态的详细信息，请见以下内容：
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 安全程序的数据类型的范围没有限制。

表 5-7: 整数数据类型

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 ~ T#+24d20h31m23s647ms

5.9.2 隐式转换

在安全型应用中，可能需要使用不同数据类型的变量来执行算术运算函数。为此所需的函数块需要形式参数的定义数据格式。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：

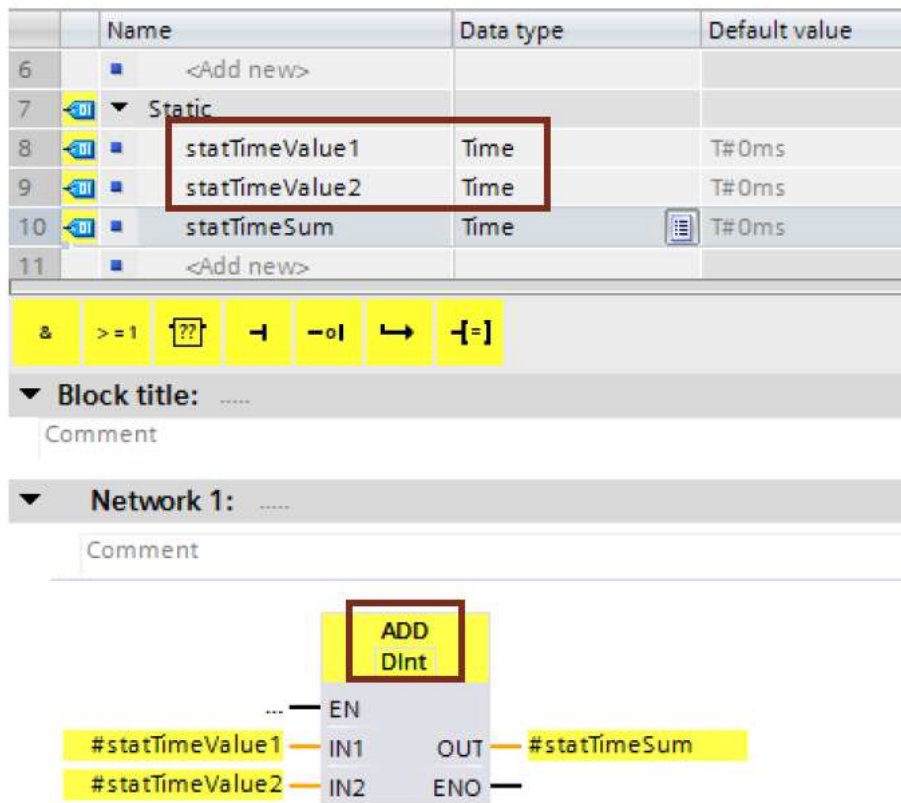
- IEC 检查被禁用。
- 数据类型具有相同的长度。

因此，可以在安全程序中隐式转换以下数据类型：

- WORD ↔ INT
- DINT ↔ TIME

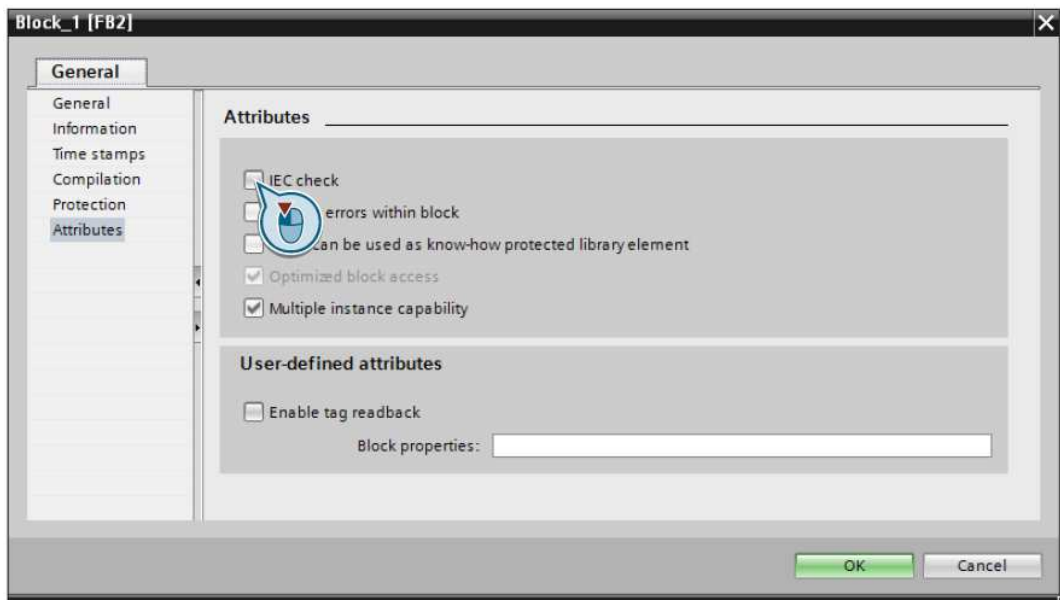
尽管函数“Add”需要作为“Dint”输入，实际应用需要两个时间值相加。结果也作为“Time”变量输出。

图 5-4: 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 5-5: 禁用 IEC 检查



5.10 符合 F 的 PLC 数据类型

对于安全程序，也可以通过 PLC 数据类型来最佳安排数据的结构。

优点

- PLC 数据类型的更改会在用户程序中的所有使用位置自动更新。

属性

- F-PLC 数据类型的声明和使用方式与 PLC 数据类型相同。
- F-PLC 数据类型可以使用安全程序中允许的所有数据类型。
- 不支持在其它 F-PLC 数据类型内嵌套 F-PLC 数据类型。
- F-PLC 数据类型可在安全程序以及标准用户程序中使用。

建议

- 要访问 I/O 区域，请使用 F-PLC 数据类型（与 [3.6.5 通过 PLC 数据类型访问 I/O 区域](#)中的情况相同）。
- 必须在遵守以下规则：
 - 符合 F 的 PLC 数据类型的变量结构必须与 F-I/O 的通道结构匹配。
 - 例如，对于带有 8 个通道的 F-I/O，符合 F 的 PLC 数据类型如下：
- 8 个 BOOL 变量（通道值），或
- 16 个 BOOL 变量（通道值+值状态）
 - 只允许针对激活的通道访问 F-I/O。在组态 1oo2 (2v2) 检测时，总是禁用更高级通道。

示例

图 5-6: 通过 F-PLC 数据类型访问 I/O 区域

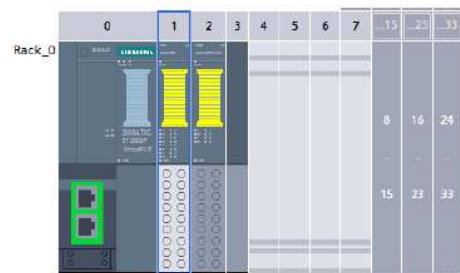
F-PLC 数据类型

typeFDIx24VDCHF	
Name	Data type
!inputCh0	Bool
!inputCh1	Bool
!inputCh2	Bool
!inputCh3	Bool
!inputCh4	Bool
!inputCh5	Bool
!inputCh6	Bool
!inputCh7	Bool
!inputCh0VS	Bool
!inputCh1VS	Bool
!inputCh2VS	Bool
!inputCh3VS	Bool
!inputCh4VS	Bool
!inputCh5VS	Bool
!inputCh6VS	Bool
!inputCh7VS	Bool

PLC tag

Name	Address
!fdi1	typeFDIx24...

F-I/O



F-DI 8x24VDC HF_1 [F-DI 8x24VDC HF]				
General	IO tags	System constants	Texts	
Name	Type	Address	Tag table	Comment
	Bool	%I4.0		!fdi1 (typeFDIx24VDCHF)
	Bool	%I4.1		!fdi1.!inputCh0
	Bool	%I4.2		!fdi1.!inputCh1
	Bool	%I4.3		!fdi1.!inputCh2
	Bool	%I4.4		!fdi1.!inputCh3
	Bool	%I4.5		!fdi1.!inputCh4
	Bool	%I4.6		!fdi1.!inputCh5
	Bool	%I4.7		!fdi1.!inputCh6
	Bool	%I4.7		!fdi1.!inputCh7

5.11 TRUE / FALSE

如果需要在安全程序中使用“TRUE”和“FALSE”信号，可能有两种情况：

- 作为块的实际参数
- 作为运算的赋值

块的实际参数

对于 S7-1200F/1500F 控制器，可以使用布尔常量“FALSE”代表 0 并使用“TRUE”代表 1，以作为用于在安全程序中的块调用期间提供形式参数的实际参数。只有关键字“FALSE”或“TRUE”写入形式参数。

图 5-7：作为实际参数的“TRUE”和“FALSE”信号



作为运算的赋值

为了给运算创建“TRUE”或“FALSE”信号，请按照以下步骤操作：

1. 创建两个 BOOL 型静态变量“statTrue”和“statFalse”。
2. 将默认值“false”赋值给变量 statFalse。
3. 将默认值“true”赋值给变量 statTrue。

可以在完整的函数块中使用变量作为“TRUE”和“FALSE”读取信号。

图 5-8：“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来保护用户编程。目的是发现安全程序中的任何类型的数据损坏，从而防止不安全的情况。

该保护程序是在编译期间创建的，因此，延长了编译时间。通过保护程序，CPU 的运行时间也延长了，因为 CPU 额外处理，并将结果与用户程序进行比较。

由系统自动生成的保护程序可以在您的 CPU 系统块文件夹中找到。

示例

图 5-9: 用户和系统创建的 F 块



本章向您列示了缩短编译和程序运行时的不同选项。

根据用途的不同，并不一定要遵循所有建议。尽管如此，这些建议仍然说明了某些编程方法比非优化程序导致更短的编译和程序运行时的原因。

5.12.1 避免时间处理块：TP, TON, TOF

每个时间处理块（TP、TON、TOF）都需要额外的块和保护代码的全局数据校正。

建议

尽可能少用这些块。

5.12.2 避免深度调用层次

深度调用层次扩大了系统创建的 F 块的代码，因为需要更大范围的保护函数和检验。当嵌套深度超过 8 时，TIA Portal 将在编译期间发出警告。

建议

避免以不必要的深度调用层次方式来安排您的程序结构。

5.12.3 避免 JMP/LABEL 结构

如果通过 JMP/LABEL 跳转一个块调用，将在系统侧的 F 块中产生额外的保护。这样，必须对跳过的块调用执行校正代码。这样将在编译过程中耗费性能和时间。

建议

尽可能地避免 JMP/LABEL 结构，以减少系统侧的 F 块。

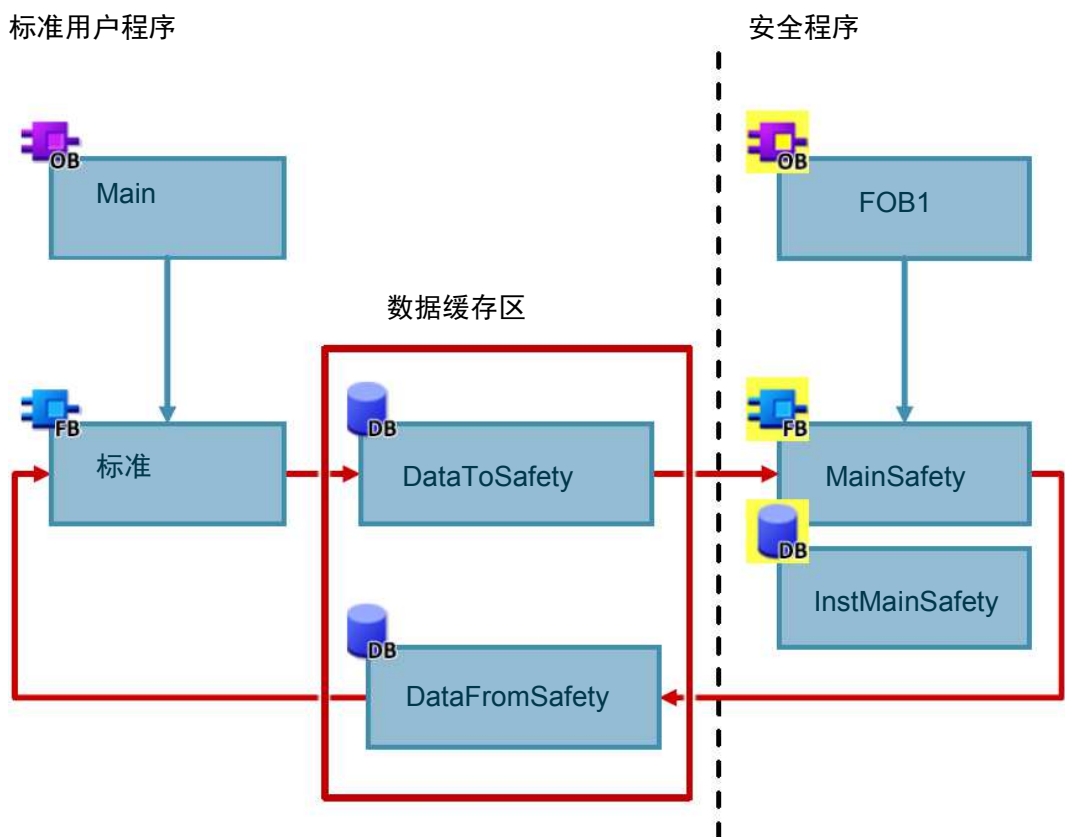
5.13 标准程序和 F 程序之间的数据交换

某些情况下，需要在安全程序与标准用户程序之间交换数据。为了保证标准程序与安全程序之间的数据一致性，必须注意以下建议。

建议

- 并不通过位存储器进行数据交换（参见 [4.2 不通过位存储器而通过全局数据块交换数据](#)）
- 将安全程序和标准用户程序之间的访问集中在两个标准数据块上。因此，标准程序的变更不会对安全程序产生影响。控制器也不需要处于 STOP 模式来加载标准程序。

图 5-10：标准程序和安全程序间的数据交换



5.14 测试安全程序

除了标准用户程序的始终可以控制的数据外，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（用于 F 运行组通信的数据块除外）、F-FB 的背景数据块
- F-I/O 数据块

属性

- 只能在 F-CPU RUN 模式下控制 F-I/O。
- 可以从监控表来控制安全程序中的最多 5 点输入/输出。
- 可以使用多个监控表。
- 需要针对“循环开始”或“循环结束”将触发点设置为“永久”(permanent) 或“一次”(once)。
- 可针对 F-I/O 执行强制。
- 如果仍希望使用停止点来进行测试，则需要事先禁用安全模式。这会导致以下错误：
 - 与 F-I/O 通信时出错
 - 故障安全 CPU-CPU 通信时出错

5.15 发生 F 错误时的 STOP 模式

以下情况下，将为 F-CPU 触发 STOP 模式：

- 在“系统块”(System blocks) 文件夹中，不能添加、更改或删除任何块。
- 不能访问未在安全程序中调用的 F-FB 的背景数据块。
- 不能超出参数“F 运行组的最大循环时间”。选择该 F 运行组的两次调用之间可以经过的“F 运行组的最大循环时间”(Maximum cycle time der F run-time group) 的最大允许时间。
- 如果从 DB 中读取变量用于 F 运行组通信，该运行组未被处理（不调用 F 运行组的主安全块）。
- 不允许在线和离线编辑 F-FB 的背景数据块中的初始值，这种编辑可导致 F-CPU 进入 STOP 模式。
- 主安全块不能包含任何参数，因为无法提供这些参数。
- 必须总要初始化 F-FC 的输出。

5.16 移植安全程序

可在以下网址获得有关移植安全程序的信息：

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

5.17 一般安全建议

以下建议通常适用于处理 STEP 7 Safety 和 F 模块。

- 应尽可能使用 F 控制器。这样，以后就可以非常简便地扩展安全功能。
- 总是要使用密码来保护安全程序，防止未经授权而对其更改。密码是在“安全管理”编辑器中设置的。

6 使用用户程序自动生成可视化

6.1 简介

从 TIA Portal V14 开始，可以使用 SiVArc (SIMATIC Visualization Architect) 选项包，从可视化库和控制器中的用户程序自动生成设备的可视化。

在本章中，您需要优化您的用户程序，以便与 SiVArc 一起使用。

使用 SiVArc 生成可视化的优势

- 自动生成过程连接可视化
- 标准化用户界面
- 对操作画面进行简单一致的调整

要求

使用 SiVArc 的基本要求是您设备实现高度标准化。将系统模块化为单个功能组的优势在于，SiVArc 可以使用这些功能组从现有的库画面生成操作画面并将其互连。界面的标准化有助于高效工作和可视化的自动生成。

遵守本编程指南的一般建议能够为您提供帮助。

注

SiVArc 是 HMI 与控制器之间的接口。本节从控制的角度来介绍 SiVArc。

访问以下链接，深入了解 SiVArc 的功能：

应用示例“SiVArc 使用入门”

<https://support.industry.siemens.com/cs/ww/en/view/109740350>

SiVArc 手册

<https://support.industry.siemens.com/cs/ww/en/view/109755214>

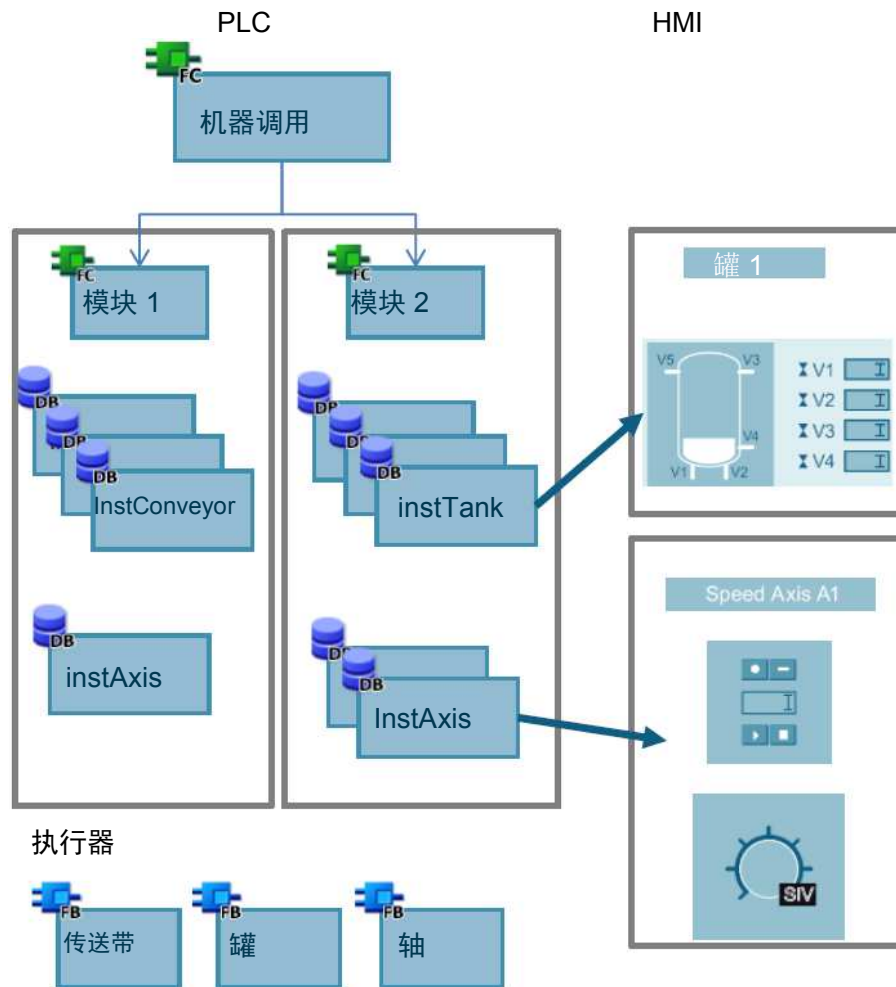
SITRAIN 培训课程：SIMATIC Visualization Architect, 自动 HMI 生成

<https://support.industry.siemens.com/cs/ww/en/view/109758628>

6.2 如何实现自动生成

在用户程序中调用标准化块（例如发动机控制块）。依照所谓的 SiVArc 规则，将被调用的块与可视化元素（文本字段、IO 字段、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 使用这些规则为指定块的每次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

Name	Program block	Screen object	Master copy of a screen	Layout field
rule	EnS_EnergyDataBasic	EnS_EnergyObjectVisualization	EnS_EnergyO...	EnS_Visu_Field

注 还可以选择限制或阻止在控制器中执行规则。

6.3 控制 HMI 生成器

选择以下控制 HMI 生成器的选项：

- 禁止为特定调用生成 SiVArc
- 根据功能或设备位置对 SiVArc 生成进行分类
- 为 SiVArc 生成添加更多属性

在 SiVArc 规则中，可以使用控制器的以下信息，以及其它信息：

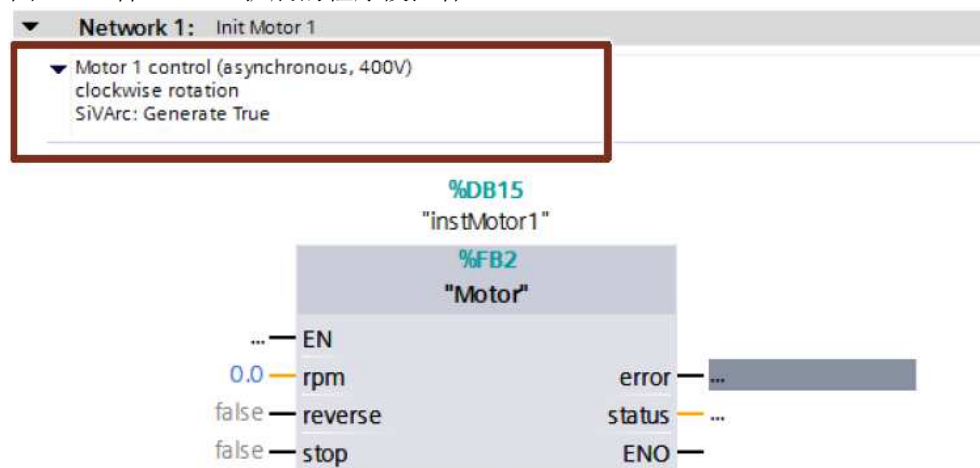
- 程序段注释
- SiVArc 变量

这样提供了在控制器中为 SiVArc 提供附加信息的选项，可以将其视为 SiVArc 规则中的一个条件。

6.3.1 使用程序段注释进行控制

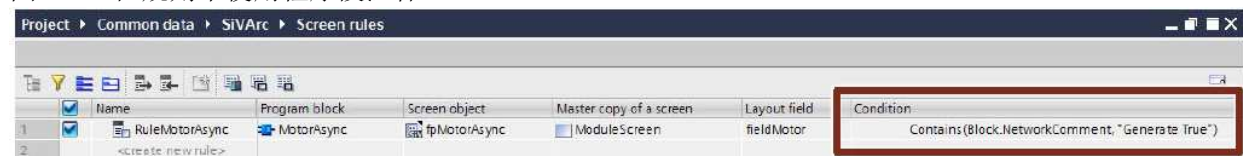
在程序段注释中，可以添加有关控制生成过程中 SiVArc 规则搜索的生成器的信息：

图 6-3：含 SiVArc 扩展的程序段注释



在规则编辑器中，使用“Condition”列中的“Contains”功能，搜索信息，例如“Contains(Block.NetworkComment, "String")”。

图 6-4：在规则中使用程序段注释



借助该功能，可使用自由设计的框架来限制规则的执行，或者只对某些程序段执行一项规则。

建议

在程序段注释中清楚地标注 SiVArc 生成的信息，例如“SiVArc:Generate True”。

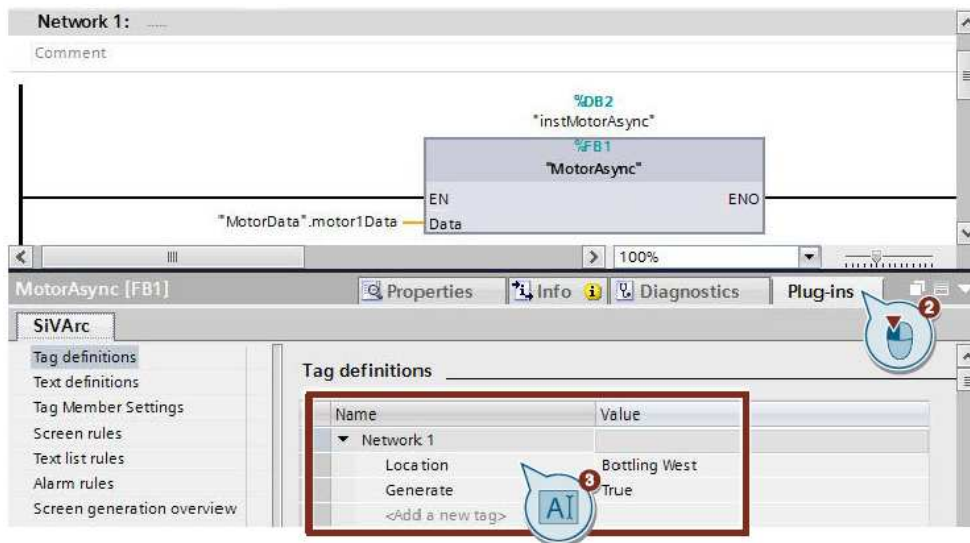
6.3.2 使用 SiVArc 变量进行控制

可以为设备中的每个程序段定义专用 SiVArc 变量，并在 SiVArc 规则中使用这些变量。

如要创建 SiVArc 变量，请执行以下操作：

4. 打开块。
5. 切换到巡视窗口中的“插件”(Plug-Ins) 选项卡。
6. 对于每个程序段，在“名称”(Name) 列中输入变量的名称，并在“值”(Value) 列中输入一个字符串值。

图 6-5: 创建 SiVArc 变量

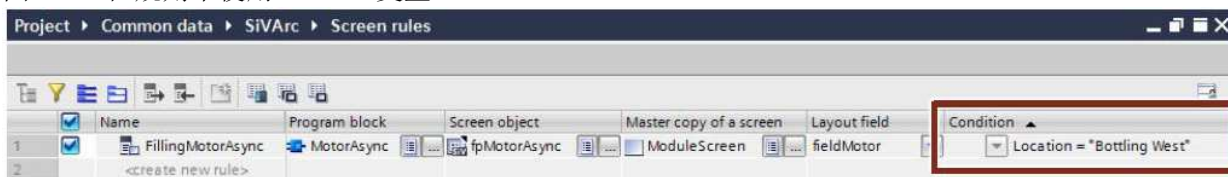


在规则编辑器中，可以使用表达式“variable name=“value””来查询 SiVArc 变量，从而
影响规则的执行。

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

在本例中，仅当 SiVArc 变量“位置”(Location) 在待生成的程序段中设置为值“Bottling
West”时，才执行该规则。

图 6-6: 在规则中使用 SiVArc 变量



建议

使用通用变量名来控制生成器，以简化规则创建。下表显示了 SiVArc 变量示例：

表 8: SiVArc 变量示例

名称	值	含义
Generate	true, false	SiVArc 为该程序段生成元素。
Location	Bottling West, Mixing etc.	可以使用此功能自动将元素分配给 HMI 图像。

注 必须使用规则编辑器中的条件定义“含义”列中列示的效果。SiVArc 变量本身对生成器没有影响。

6.4 其它建议

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、变量或程序段注释的标识符。SiVArc 会删除 WinCC 不支持的所有字符。

这样可导致项目中出现不一致。

不支持的字符包括：

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

使用编程语言 FBD 调用块

因此，可以通过程序段注释或 SiVArc 变量来控制生成器，使用编程语言 FBD 调用从中生成可视化元素的块。

注 该项建议适用于 TIA Portal V15 及以下版本。
从 TIA Portal V15.1 开始，也可以使用 SCL 模块。

7

最重要的建议

- 使用优化的块
 - [2.6 优化的块](#)
- 使用数据类型 **VARIANT** 来替代 **ANY**
 - [2.8.5 数据类型 VARIANT](#)
- 清晰有序地安排程序结构
 - [3.2 程序块](#)
- 将指令作为多重实例插入 (TON、TOF ..)
 - [3.2.5 多重实例](#)
- 可重复使用的块编程
 - [3.2.9 块的可再用性](#)
- 符号编程
 - [3.6 符号寻址](#)
- 使用 **ARRAY** 来处理数据
 - [3.6.2 ARRAY 数据类型和间接域访问](#)
- 创建 **PLC** 数据类型
 - [3.6.5 通过 PLC 数据类型访问 I/O 区域](#)
- 使用库来存储程序元素
 - [3.7 库](#)
- 不使用位存储器而使用全局数据块
 - [4.2 不使用位存储器而使用全局数据块](#)

8 附录

8.1 服务与支持

工业业务领域在线支持

欢迎提出宝贵的意见和建议！

西门子工业在线支持可全天候提供维护、专有技术和产品组合服务与支持。

工业在线支持是有关西门子产品、解决方案和服务的信息中心。

包括产品信息、手册、下载、常见问题、应用示例和视频。

– 只需点击几下鼠标，即可了解所有信息：<https://support.industry.siemens.com>

技术支持

西门子工业技术支持以众多定制方案为您提供所有技术查询的快速、高效支持：

– 从基本支持到具体支持联络人。请通过 **Web** 发送咨询到技术支持中心：

www.siemens.com/industry/supportrequest

SITRAIN – 工业培训

我们将针对工业领域提供全球培训课程，其中包括实际操作、创新的学习方法以及可根据客户的特定需求进行定制。

有关我们提供的培训和课程及其位置和日期的更多信息，请参见网址：

www.siemens.com/sitrain

服务

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

您可以在服务样本网页上找到有关我们服务范围的详细信息：

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

“工业业务领域在线支持”应用

无论您身在何处，都可以通过“Siemens Industry Online Support”应用获得最佳支持。

该应用可用于 Apple iOS、Android 和 Windows Phone：

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

8.2 链接与文献

表 8-1: 链接与文献

	主题
\1\	西门子工业在线支持 https://support.industry.siemens.com
\2\	下载中心 https://support.industry.siemens.com/cs/ww/en/view/81318674
\3\	S7-1200 和 S7-1500 编程风格指南 https://support.industry.siemens.com/cs/ww/en/view/81318674
\4\	STEP 7 (TIA Portal) 和 S7-1200/S7-1500 (LGF) 的通用函数库 https://support.industry.siemens.com/cs/ww/en/view/109479728
\5\	STEP 7 (TIA Portal) 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 Portal – 最重要的文档和链接概览 https://support.industry.siemens.com/cs/ww/en/view/65601780
\7\	STEP 7 (TIA Portal) 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 CPU 手册 https://support.industry.siemens.com/cs/ww/en/ps/13888/man
\11\	S7-1200 使用入门 https://support.industry.siemens.com/cs/ww/en/view/39644875
\12\	S7-1500 使用入门 https://support.industry.siemens.com/cs/ww/en/view/78027451
\13\	基于国际助记法的 SIMATIC S7-1200/S7-1500 编程语言对照表 https://support.industry.siemens.com/cs/ww/en/view/86630375

8.3

文档变更

表 8-2: 文档变更

版本	日期	修改
V1.0	09/2013	第一版
V1.1	10/2013	以下章节的更正内容： 2.6.3 S7-1500 的处理器优化数据存储 2.12 用户常量 函数 (FC) 函数块 (FB) 3.4.3 局部存储器
V1.2	03/2014	新章节： 2.6.4 优化和非优化的变量之间的转换 2.6.6 使用优化数据进行通信 2.9.1 MOVE 指令 2.9.2 VARIANT 指令 3.6.5 通过 PLC 数据类型访问 I/O 区域 以下章节的更正内容： 2.2 术语 2.3 编程语言 2.6 优化的块 2.10 符号与注释 3.2 程序块 3.5 保持性 4.3 “时钟位”编程 不同章节中的各种连接
V1.3	09/2014	新章节： 2.8.4 Unicode 数据类型 2.10.2 监控表中的注释行 2.12 用户常量 3.2.10 块的自动编号 5 TIA Portal 中的 STEP 7 Safety 以下章节的更正内容： 2.7 块属性 S7-1200/1500 的数据类型 指令 3.6.4 STRUCT 数据类型和 PLC 数据类型 3.7 库 不同章节中的各种连接

版本	日期	修改
V1.4	11/2015	新章节： 2.6.5 优化和非优化访问块之间的参数传输 3.3.3 参数传送一览 3.10.5 正确使用 FOR、REPEAT 和 WHILE 循环 5.12 优化编译和程序运行时间
V1.5	03/2017	新章节： <u>2.7.3 Block interface – hide block parameters (V14 or higher)</u> 2.9.4 PLC 数据类型变量的比较 (V14 或更高版本) 2.9.5 <u>多重赋值 (V14 或更高版本)</u> <u>3.2.6 以参数形式传送实例 (V14)</u> 3.6.3 形式参数 Array[*] (V14 或更高版本) 3.6.7_LAD 和 FBD 中的 SCL 程序段 (V14 及更高版本) 3.10.4 使用关键字 REGION (V14 或更高版本) 结构化 3.10.11 不必要的 IF 指令 在不同章节中进行了多项更正
V1.6	12/2018	新章节： 6 使用用户程序自动生成可视化 更新了标题页和法律信息