程序设计规范指南

适用于 SIMATIC S7-1200/S7-1500

TIA Portal

https://support.industry.siemens.com/cs/ww/en/view/109478084

西门子
工业技术
支持中心

法律信息

应用实例的使用

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

免责声明

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

通过使用应用实例，您承认西门子不承担超出上述责任条款的任何损害。

其他信息

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

安全信息

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

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

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

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

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

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

目录

1 导言

SIMATIC 控制器编程人员的任务是开发尽可能易读和结构化的应用程序。 每个开发人员有自己的策略来实现这个任务，例如变量、块的命名或程序的注释方式。 不同的开发人员使用不同的习惯，因此存在很多不同的程序风格，这些程序通常只能由各自的创建者来说明。

注意: 本文档是基于 SIMATIC S7-1200/S7-1500 的编程指南，其中描述了控制器 S7-1200 和 S7-1500 如何以最佳方式对它们进行编程:

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

1.1 目标

以下章节中描述的规则和建议可以帮助您创建一个统一的、可维护和可重用的程序代码。特别在多个开发人员共同开发的情况下， 建议规定项目范围内的术语以及统一的编程风格。通过这种方式，您可以在项目早期阶段检测并避免错误。

出于可维护性和可读性的考虑，程序需要遵循一定的格式，更重要的是定义规则，这些规则可为开发人员带来如下益处：

- 避免编译器无法识别的错误，例如错别字等不经意的错误
  目标：编译器应识别尽可能多的错误
- 支持开发人员诊断编程错误，例如临时变量的重复使用超过一个周期
  目标：尽早指出标识的问题
- 应用程序和库的标准化
  目的：使培训变得容易，增加程序代码的可重用性
- 易于维护和简化下一步开发
  目标：因为程序的更改可能是由不同的程序员在执行，所以要求在程序代码单个模块中所做的更改，应该对整个程序产生最小的影响

注意: 本文件中所描述的规则和建议是一致的，互不干扰。

1.2 统一规则的优点

- 风格统一一致
- 易于阅读和理解
- 易于维护，可重用性强
- 简单快速的错误识别和纠正
- 多个程序员的高效协作

1.3 适用性

本文件适用于 TIA 博途中的项目和库，这些项目和库是使用符合 IEC 61131-3 的编程语言编写的（DIN EN 61131-3），它们是结构化文本（SCL/ST），梯形图（LAD/KOP）和函数块图（FBD/FUP）。

本文件还适用于软件单元，文件夹，组，组织块(OB)，函数(FC)，函数块(FB)，工艺对象(TO)，数据块(DB)，PLC 数据类型(UDT)，变量，常量，PLC 消息文本列表，监视表，强制表以及外部源。

1.4 范围

此文档不包含以下内容的描述：

- 使用 TIA 博途的 STEP 7 编程
- SIMATIC 控制器的调试

在以上内容有足够的知识和经验，是正确理解和使用既定规则及建议的先决条件。

本文档仅作为参考，并不取代软件开发领域的相关知识。

1.5 违规及其他规则

在客户项目中，应遵循适用的规则、客户或行业特定标准以及技术规则（如安全、运动控制等），并优先于此风格指南或部分使用。

当把客户规则与本风格指南中的规则相结合时，必须特别注意保持规则的完整性和一致性。任何违反规则的行为都必须有正当理由并适当记录在用户程序中。
客户提供的规章制度必须形成相应的记录。

2 定义

2.1 规则/建议

本文件中的规定包含两个部分，一是建议，二是规则：

- **规则**有一定约束力，必须遵循， 它们对于可重用编程和高效编程是必不可少的。 在特殊情况下，可能会违反规则，这种情况下必须证明是合理的并记录在案。
- **建议**是规定，它支持程序代码的统一性，并起到支持和成档的作用， 总体上应遵循各项建议。 但是，也有例外情况，为了更加高效或更佳的可读性，可能不遵守这一建议。

2.2 枚举规则

对于唯一的规则标识，在类别内，规则和建议用前缀（2 个字符）标识，并设置枚举（3 位数字）。

如果规则被取消，其编号将不会被重新分配；如果需要更多的规则，您可以使用 901 到 999 之间的数字。

表 2-1

前缀	类别
ES	Engineering System: 程序设计环境
GL	Globalization: 全球化
NF	Nomenclature and formatting: 命名和格式
RU	Reusability: 可重用性
AL	Allocation: 对象的引用
SE	Security: 安全
DA	Design and architecture: 设计和体系结构
PE	Performance: 性能

2.3 性能

自动化系统的性能是由程序的执行时间来定义的。

当提到性能损失时，这意味着可以通过应用编程规则和有效的编程方式来减少程序执行时间，因此用户编程的周期时间会有所增加。

2.4 标识符/命名

区分标识符和名称很重要，名称是标识符的一部分，它描述了标识符的含义。
标识符由以下各项组成:

- 前缀
- 名称
- 后缀

2.5 缩略语

本文件通篇使用以下缩略语:

表 2-2

缩写	类别
OB	组织块
FB	函数块
FC	函数
DB	数据块
TO	工艺对象
UDT	PLC 数据类型

2.6 与变量和参数一起使用的术语

当涉及到变量，函数和函数块时，有很多术语，这些术语在不同场景中很可能会被错误的使用，所以确保对本文档中术语的一致理解是非常必要的。下图会说明这些术语。

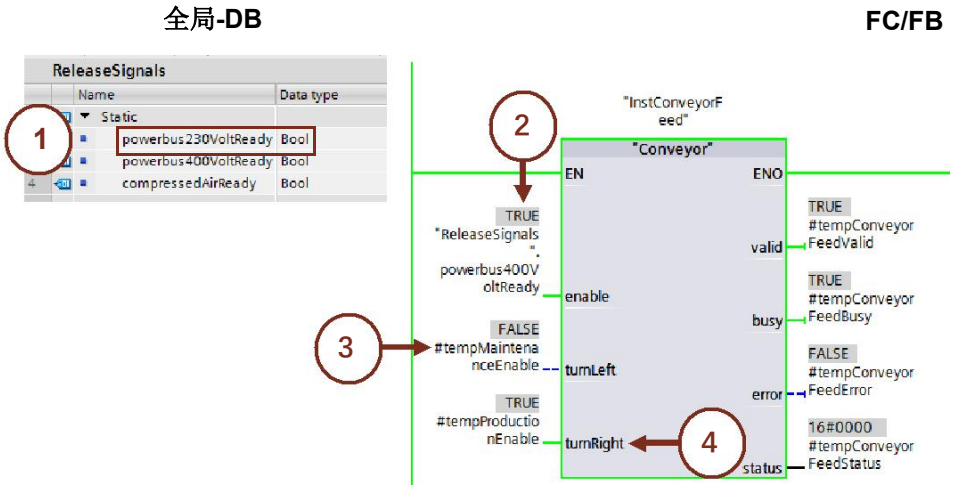

图 2-1

表 2-3

	术语	描述
1	变量	变量由标识符声明，并在控制器的内存中分配一个地址。 变量总是以某种数据类型（布尔、整数等）定义： • PLC 变量或用户常量 • 块中的变量或常量 • 结构变量（“STRUCT”）和 PLC 数据类型 • 数据块/背景数据块 • 工艺对象
2	当前值 实际值	当前值是存储在变量中的值(例如，15 作为整数变量的值)
3	实际参数	实际参数是与块的形式参数连接的变量
4	形式参数	形式参数是在块的接口中声明的变量，用于程序调用。 形式参数通常被称为“接口参数“或”传递参数“，但是应该避免这些术语。

块接口由形式参数和本地数据两部分组成。

形式参数

表 2-4

类型	区域	功能
输入参数	Input	参数，块从中读取值
输出参数	Output	参数，块将值写入其中
输入/输出参数	InOut	参数，块从中读取值，处理值并将处理后的值写回相同的参数
返回值	Return	值，返回到调用块

局部数据

表 2-5

类型	区域	功能
临时变量	Temp	存储中间值的变量
静态变量	Static	存储持久/静态中间变量值到实例数据块中
常量	Constant	具有符号标识符的常量值在单个块内使用

3 TIA 博途中的设置

在本章中描述了编程环境初始化设置的规则和建议。

注意： 此处列出的 TIA 博途中设置的规则和建议存储在 TIA 博途设置文件（tps 文件）中， 您可以找到单独下载的 tps 文件用于本章节。 要应用这些设置，您可以将 tps 文件导入 TIA 博途。

ES001 规则：用户界面语言“English”

用户界面语言设置为“English”， 保证所有新创建的项目的编辑和参考语言，所有系统常量都设置为英语。

说明：如果要求所有系统常量都可用同一种语言，用户界面语言必须设置为一种通用的统一语言。

ES002 规则：助记符“International”

助记符（编程语言的语言设置）应设置为“International”。

说明：所有的系统语言和系统参数都是系统独立设置的，保证团队中的编程人员之间能够进行无缝的合作。

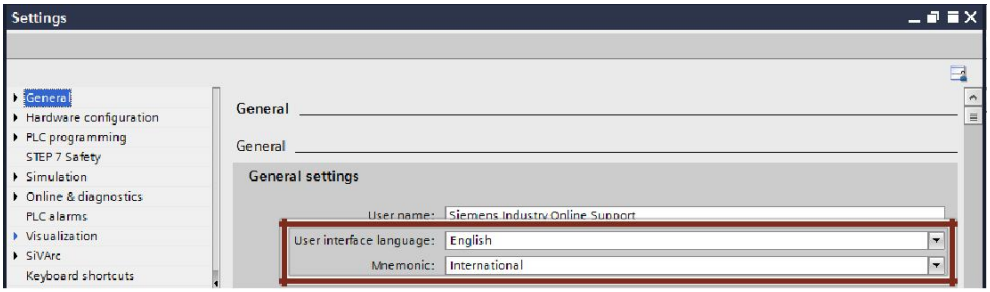

图 3-1

ES003 建议：编辑器中的非比例字体

对于编辑器，建议使用非比例字体（等宽字体）。 所有字符具有相同的宽度并且代码、单词和缩进是统一的。

建议设置的字体为“Consolas”，大小 10 pt。

说明: 与带有“Consolas”的“Courier New”相比，重点是相似字符之间的区别。它是专门为编程环境设计的。

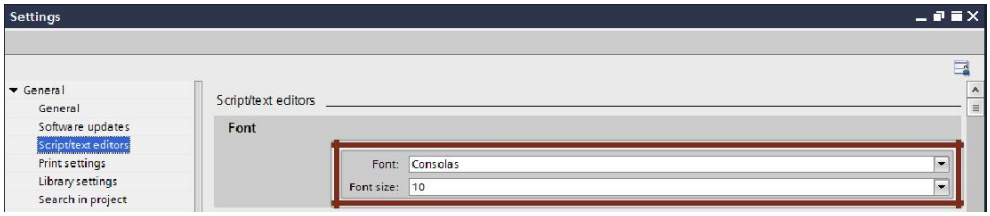

图 3-2

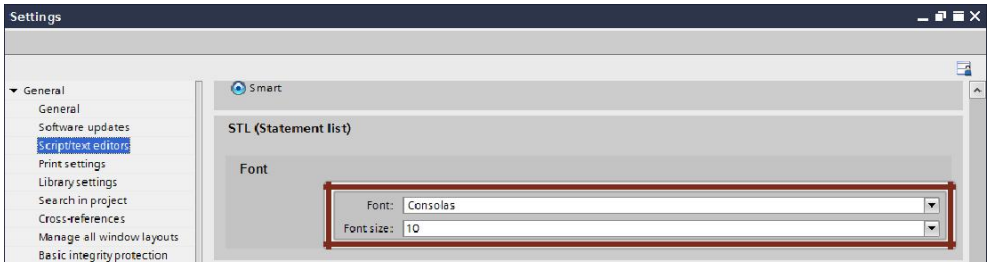

图 3-3

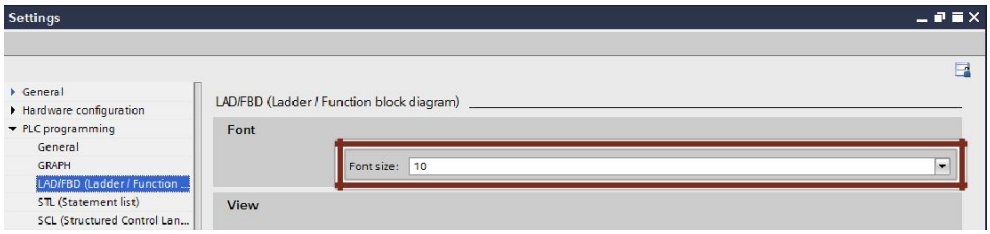

图 3-4

ES004 规则：带有两个空格的智能缩进

对于指令的缩进，使用两个空格。 可将选项中的“Indent”设置更改为“Smart”。 在基于文本的编辑器中不允许使用制表符，因为其宽度的显示方式不同。

说明:使用此设置，即使使用不同的编辑器，也可以提供统一的显示样式。

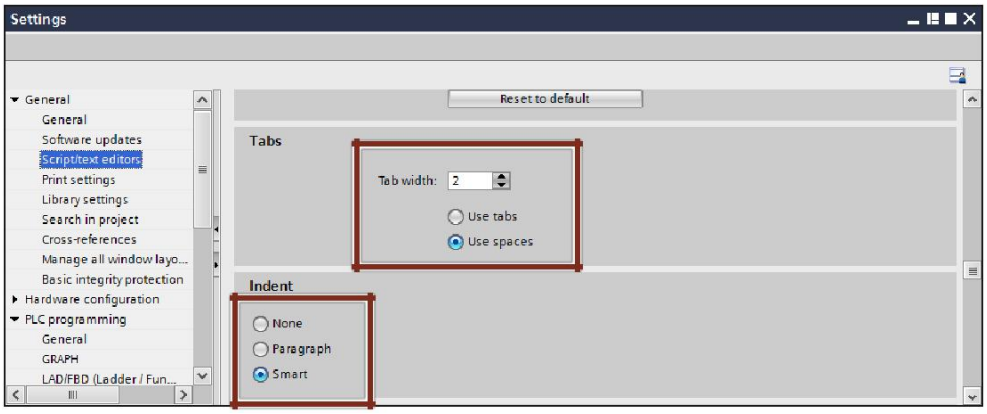

图 3-5

ES005 规则：操作数的符号表示

操作数的表示设置为“Symbolic”。

说明:程序设计完全基于符号。

图 3-6

ES006 规则：符合 IEC 的编程

为了遵守 IEC 编程，默认设置为每个新程序块打开 IEC 检查。

说明:启用设置后，每个新的程序块都打开了 IEC 检查，这确保了变量的类型安全使用。

图 3-7

ES007 规则：通过 HMI/OPC UA/Web API 显式数据访问

从 HMI/OPC UA/Web API 禁用接口的可访问性和可写性限制了外部应用程序对内部数据的访问。

说明:外部应用程序应该只能在配置显示启用时访问内部数据。

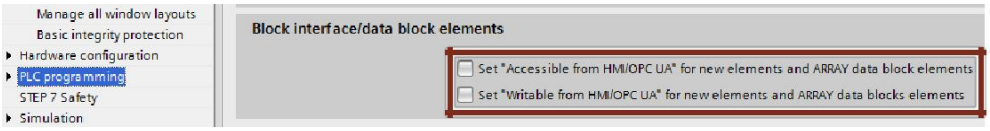

图 3-8

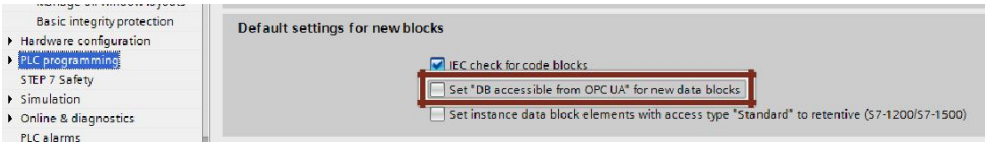

图 3-9

ES008 规则：启用自动值计算(ENO)

EN/ENO 机制用于对类型定义的值边界及其操作的自动计算。 默认情况下，此机制处于启用状态。

说明: 激活此设置后，ENO 评估由系统执行，另请参阅“SE003 规则：处理 ENO”。

ES009 规则：自动计算数组边界

必须打开数组边界的自动计算。

说明:编译时检查数组边界，可以避免越界访问。

4 全球化

本章介绍用于全球合作的规则和建议。

GL001 规则：使用一致的语言

PLC 和 HMI 编程中使用的语言必须一致。 这意味着，英语文本只能在英语语言设置中找到。

GL002 规则：设置编辑参考语言为“English（US）”

如果客户没有其他要求，则编辑语言和参考语言都必须设置为“English（US）”。包括所有注释整个程序必须用英语创建。

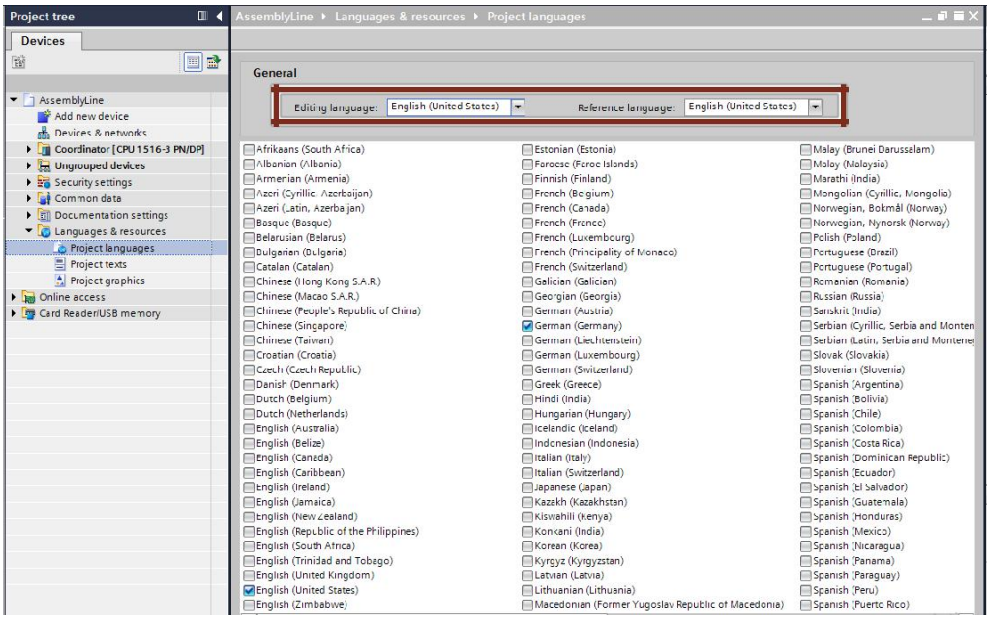

图 4-1

GL003 规则：提供所有项目语言的文本

所有项目文本至少必须提供英语，也可提供其他项目语言。

注意: 在块编辑器中，文本及其翻译可以很容易在选项卡“Texts”中进行管理。

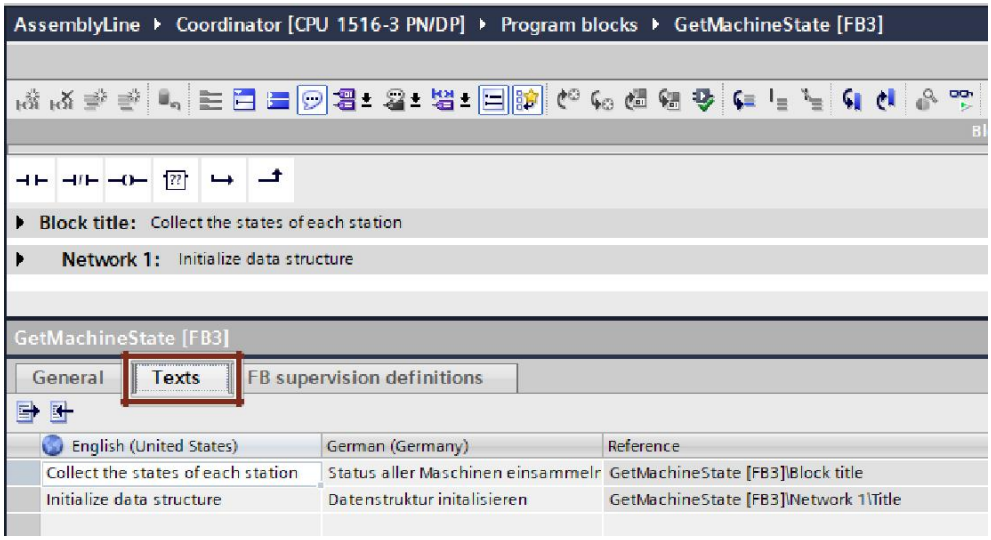

图 4-2

5 命名和格式

本章描述了命名和编写程序和注释的规则和建议。

NF001 规则：唯一且一致的英文标识符

标识符（块、变量等）的名称必须使用英语（英语-美国）。 名称描述了标识符在源代码上下文中的含义，故特此加强了对标识符功能和用法的理解。

- 在所有函数块和 PLC 数据类型中相同的标识符需保持名称的一致性，并且应尽可能简短
- 相同功能的标识符具有相同的标识符名称，这也适用于大写字母
- 标识符名称可以由多个单词组合而成，单词的顺序必须与口语中的顺序相同
- 函数(FC)和函数块(FB)标识符应以动词开头，例如“Get”，“Set”，“Put”，“Find”，“Search”，“Calc”
- 如果使用标识符给数组命名，则名称使用复数，不可数名词保持其单数形式（“data”，“information”，“content”，“management”）
- 静态和临时布尔变量往往是状态指示变量，在这种情况下，以“is”，“can”或“has”开头的名称最容易理解

说明：提供程序和其输入输出的快速概览

注意: TIA 博途提供的形参是占位符，需要被用户定义的命名替代。

表 5-1

	正确的命名	不正确的命名
数组	data beltConveyors	datas beltConveyor
静态和临时 布尔变量 用于表示状态	isConnected canScan	connected scan
其他布尔变量	enable	setEnable
函数（FC）/函数块（FB）	GetMachineState SearchDevices	MachineStateFC FB_Device

NF002 规则：使用有意义的注释和属性

注释和属性字段应使用有意义的注释描述和信息， 这包括：

- 块标题和块注释（同时参见“NF003 规则：记录开发人员信息”）
- 块接口
- 网络标题与网络注释
- 块及其变量和常量
- PLC 数据类型及其变量
- PLC 变量表，PLC 变量和用户常量
- PLC 报警文本列表
- PLC 监控和报警
- 库属性

说明: 使用此方法，用户可以在使用组件时获得更多的信息和指导，例如通过工具获取到相关提示。

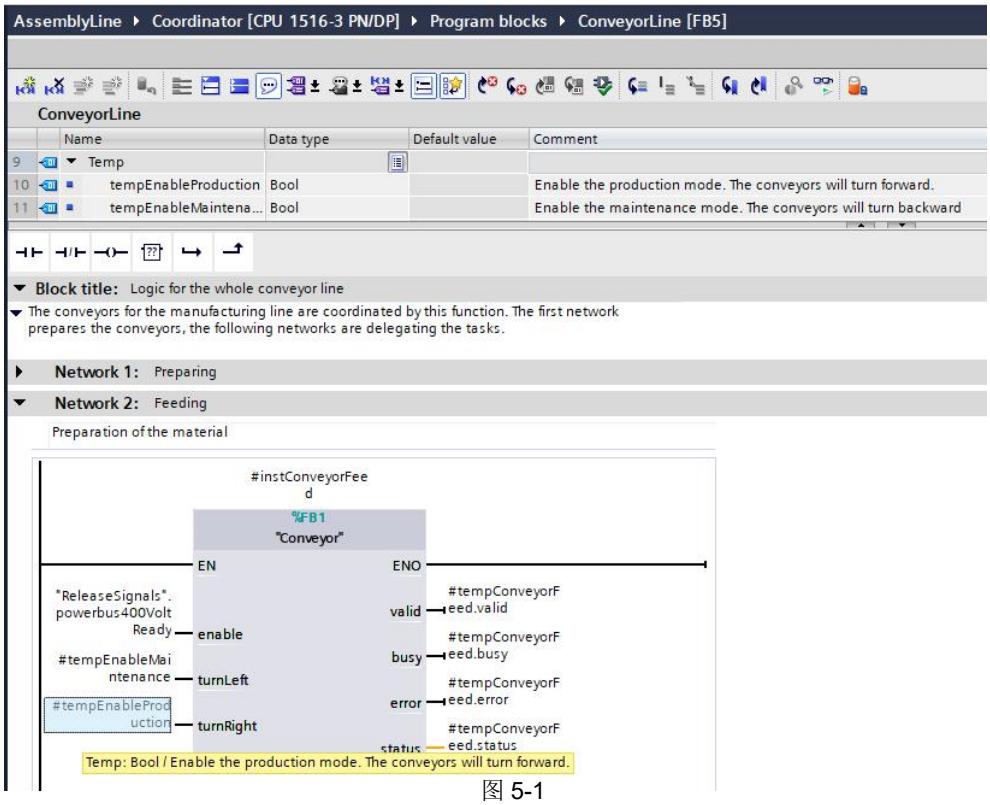

图 5-1

注意： 此外，块描述和文档可以作为博途用户自定义帮助文档（例如*.pdf，*.html），用户可以通过按<Shift><F1>打开作为联机帮助的一部分。

更多信息请在在线帮助中输入关键词“用户自定义文档“查询:

https://support.industry.siemens.com/cs/cn/zh/view/109773506/119453612171

NF003 规则：记录开发人员信息

每个块在程序代码（SCL/ST）或块注释（LAD，FBD）中包含一个块标题栏，开发过程中最重要信息必须记录在案。开发相关信息将被存放于程序内部，隐藏在专有技术保护块中。

必须在块属性中提供用户相关信息。 即使在专有技术保护的块中，用户也可以获得此信息。

下面这个块标题栏的模板包含块属性中的元素以及与开发相关的信息，它们不需要复制到属性中。

模板描述包含以下项目:

- （可选）公司名称/(C)版权（年份）版权所有
- 标题/块描述
- 功能说明
- （可选）库的名称
- 部门/作者/联系人
- 目标系统 - 带固件版本的 PLC（例如，1516-3 PN/DP v2.6)
- 工程环境 - TIA 博途，包含创建/修改时的博途版本
- 使用限制（例如特定的 OB 类型）
- 要求（如附加硬件）
- （可选）其他信息
- （可选）包含版本、日期、作者和修改说明的修改日志（对于安全块则包含安全签名）

LAD/KOP 和 FBD/FUP 中块标题的模板:

```
(company) / (C) Copyright (year)
-------------------------------------------------------------------------------
Title:            (Title of this block)
Comment/Function: (that is implemented in the block)
Library/Family:   (that the source is dedicated to)
Author:           (department / person in charge / contact)
Target System:    (test system with FW version)
Engineering:      TIA Portal (SW version)
Restrictions:     (OB types, etc.)
Requirements:     (hardware, technological package, etc.)
-------------------------------------------------------------------------------
Change log table:
Version     | Date       | Signature  | Expert in charge | Changes applied
------------|------------|------------|------------------|----------------------

001.000.000 | yyyy-mm-dd | 0x47F78CC1 | (name of expert) | First released version
```

SCL 中块标题栏的模板:

```
REGION Description header
  //===========================================================================================
  // (company) / (C) Copyright (year)
  //-------------------------------------------------------------------------------------------
  // Title:            (Title of this block)
  // Comment/Function: (that is implemented in the block)
  // Library/Family:   (that the source is dedicated to)
  // Author:           (department / person in charge / contact)
  // Target System:    (test system with FW version)
  // Engineering:      TIA Portal (SW version)
  // Restrictions:     (OB types, etc.)
  // Requirements:     (hardware, technological package, etc.)
  //-------------------------------------------------------------------------------------------
  // Change log table:
  // Version     | Date       | Expert in charge | Changes applied
  //-------------|------------|------------------|-----------------------------------
  // 001.000.000 | yyyy-mm-dd | (name of expert) | First released version
  //===========================================================================================
END_REGION
```

NF004 规则：遵守库的前缀和结构

库的标识符具有前缀“L”，并且最大长度不超过八个字符

库的标识符以前缀“L”开头，后跟最多七个字符作为名称（例如 LGF，LCom）。“L”代表单词库(Library)。 在库标识符之后使用下划线(_)作为分隔符（例如 LGF_）。

库标识符的最大长度（包括前缀）限制为八个字符。

说明:这一限制的目的是保证其名称紧凑简短。

库中的每个元素都带有前缀

库中包含的所有类型和主副本都被赋予了库的标识符。

一个用于演示的库使用的元素不是标准化库意义上的库元素，它只是一个示例，因此不一定带有库前缀。

说明: 在标识符中包含前缀的情况下可避免命名冲突。

表 5-2

类型	符合风格指南的标识符
Library，Library 的主文件夹	LExample
PLC 数据类型	LExample_type<Name>
函数块	LExampleL<Name>
函数	LExampleL<Name>
全局数据块	LExampleL<Name>
组织块	LExampleL<Name>
PLC 符号表	LExampleL<Name>
全局常量	LExampleL<NAME>

类型	符合风格指南的标识符
表示错误的全局常量	LEXAMPLE_ERROR_<NAME> LEXAMPLE_ERR_<NAME>
表示警告的全局常量	LEXAMPLE_WARNING_<NAME> LEXAMPLE_WARN_<NAME>
PLC 报警信息文本列表	LExample_<NAME>

库中的分组

所有的主副本和类型应放在库内的一个子文件夹中，该子文件夹携带库标识符作为其文件夹名。

说明: 子文件夹支持项目协同工作，并允许同一项目中有多个库的分组。

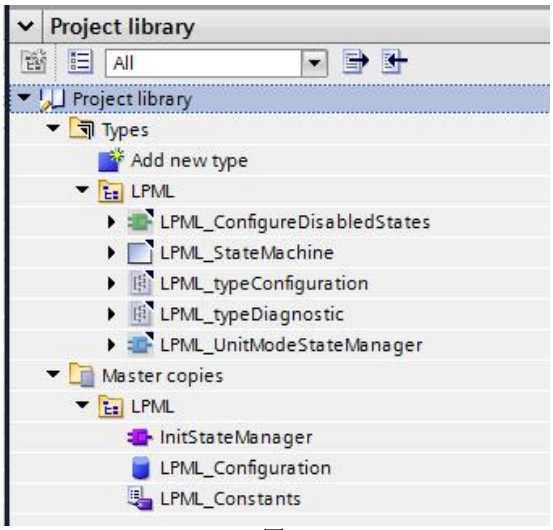

图 5-2

注意： 此规则仅包含库元素命名的信息。 此外，建议遵循上述建议及库指南中的详细信息:

https://support.industry.siemens.com/cs/ww/en/view/109747503

NF005 规则：使用帕斯卡（PascalCasing）命名法

TIA 博途对象的标识符，例如:

- 块
- 软件单元，工艺对象，库，项目
- PLC 变量表
- PLC 报警文本列表
- 监控与强制表
- 轨迹和测量

都使用帕斯卡命名法编写。

帕斯卡命名使用以下规则：

- 第一个字符是大写字母
- 如果一个标识符是由多个单词组成的，那么每个单词的第一个字符是大写字母
- 对于单一概念的标识符的命名不使用分隔符（例如连字符或下划线）；为保证结构化和专业化，允许使用少量下划线（但不超过三个）

表 5-3

合适	过多
GetAxisData_POSAxis GetAxisData_SpeedAxis GetAxisData_Syncaxis	Get_Axis_Data_Pos_Axis

NF006 规则：对代码元素使用驼峰命名规则 camelCasing

代码元素的标识符，如

- 变量
- PLC 数据类型
- 结构（“STRUCT”）
- PLC 变量
- 参数

都是使用驼峰规则。

驼峰规则使用以下规则:

- 第一个字符是非大写（小写）字母
- 如果一个标识符是由多个单词组成的，则后面单词的第一个字符大写
- 不允许使用分隔符（如连字符或下划线）进行分隔

Conveyor

	Name	Datentyp
1	Input	
2	enable	Bool
3	turnLeft	Bool
4	turnRight	Bool
5	Output	
6	valid	Bool
7	busy	Bool
8	error	Bool
9	status	Word

图 5-3

NF007 规则：使用前缀

形参不加前缀

函数块的形参不带前缀， 传递 PLC 数据类型时，标识符也不带前缀。

临时变量和静态变量带有前缀“temp”或“stat”

为了在代码中区分临时变量和形参，详情参见表 5-4 。

说明:这个措施使得程序员能够更容易区分形参和局部数据。 有了这个前缀，就可以很容易地定义和识别对变量的访问。

注意： 全局数据块和数组数据块中的静态变量不加前缀 “stat”。

前缀为“inst”或“Inst”的实例数据

单个实例以及多重实例和参数实例都有一个前缀。 单个实例的前缀为“Inst”，而多重实例和参数实例使用前缀 “inst”。

说明:有了前缀就更容易识别是否已对实例数据进行了有效（无效）访问。

前缀为“type”的 PLC 数据类型

对于 PLC 数据类型，前缀可设置为“type”，需要注意的是， PLC 数据类型中的元素不加前缀。

表 5-4

前缀	类型
无	**输入输出参数** 可以从外部访问 ➔ enable ➔ error
无	**InOut 参数** 用户和块均可对数据进行修改 ➔ conveyorAxis
无	**PLC 变量和用户常量** ➔lightBarrierLeft（PLC 变量的标识符） ➔MAX_BELTS（用户常量的标识符）
无	**全局数据块** 全局数据块和包含的元素都没有前缀 ➔ReleaseSignals（全局数据块的标识符） ➔powerBusReady（全局数据块中变量的标识符）
temp	**临时变量** 无法从外部访问本地数据 ➔tempIndex
stat	**静态变量** 不允许从外部访问本地数据 ➔statState
inst	**多重实例和参数实例的变量** ➔instWatchDogTimer ➔instWatchDogTimer**s**（带有实例数组）
Inst	**单个实例数据块** ➔InstConveyorFeed
type	**PLC 数据类型** 只有数据类型加前缀，元素不加前缀 ➔typeDiagnostic（PLC 数据类型的标识符） ➔stateNumber（PLC 数据类型内变量的标识符）

NF008 规则：用大写字母表示常量标识符

常量（全局和局部常量）的名称完全用大写字母， 为了区分和识别单个单词或缩写应该在单词或缩写中间添加下划线。

图 5-4 FB 块中的常量

注意： “TRUE”和“FALSE”也是常量。

NF009 规则：限制标识符的字符集

对于所有对象和代码标识符，只使用拉丁字母（a-z，A-Z）和阿拉伯数字（0-9）以及下划线“_”。

表 5-5

正确命名	错误命名
tempMaxLength	temporary Variable 1

NF010 建议：限制标识符的长度

标识符的总长度，包括前缀、后缀或库标识符，不得超过 24 个字符。

说明: 因为结构中的变量名是由许多标识符组成，所以代码位置处标识符的长度经常都会过长。

示例:

```
instFeedConveyor024.releaseTransportSect1.gappingTimeLeft
```

NF011 建议：每个标识符仅使用一个缩写

为了实现最佳可读性，多个缩写不得连续使用。 为了减少标识符中使用的字符量，建议的常用缩写见表 5-6。

本表仅包含最常用的缩略语， 缩略语的拼写必须遵循特定用途的规则，并需要相应地采用大小写。

表 5-6

Abbrev.	Type
Min	Minimum
Max	Maximum
Act	Actual, Current
Next	Next value
Prev	Previous value
Avg	Average
Sum	Total sum
Diff	Difference
Cnt	Count
Len	Length
Pos	Position
Ris	Rising edge
Fal	Falling edge
Old	Old value (e. g. for edge detection)
Sim	Simulated
Dir	Direction
Err	Error
Warn	Warning
Cmd	Command
Addr	Address

NF012 规则：以对应的格式初始化变量

变量的初始化（常量数据的赋值）应符合适当的格式，如一个 WORD 类型的变量应该用 16#0001 初始化而非 16#01。

在代码中完成的初始化应使用局部符号常量，也请参考“RU005 规则:使用局部符号常量”。

表 5-7

正确初始化			错误初始化		
statTriggerOld	Bool	FALSE	statTriggerOld	Bool	false
statStatus	Word	16#7000	statStatus	Word	123
statStep	DInt	101	statStep	DInt	16#0
statVelocity	LReal	0.0	statVelocity	LReal	16#000
statCommand	Byte	16#01	statCommand	Byte	16#1
statFlags	Byte	2#1010_0101	statFlags	Byte	25

NF013 建议：隐藏可选形参

隐藏可选的形参。

说明: 通过折叠可选的形参，这样可以将块调用减少到必要的最小值。

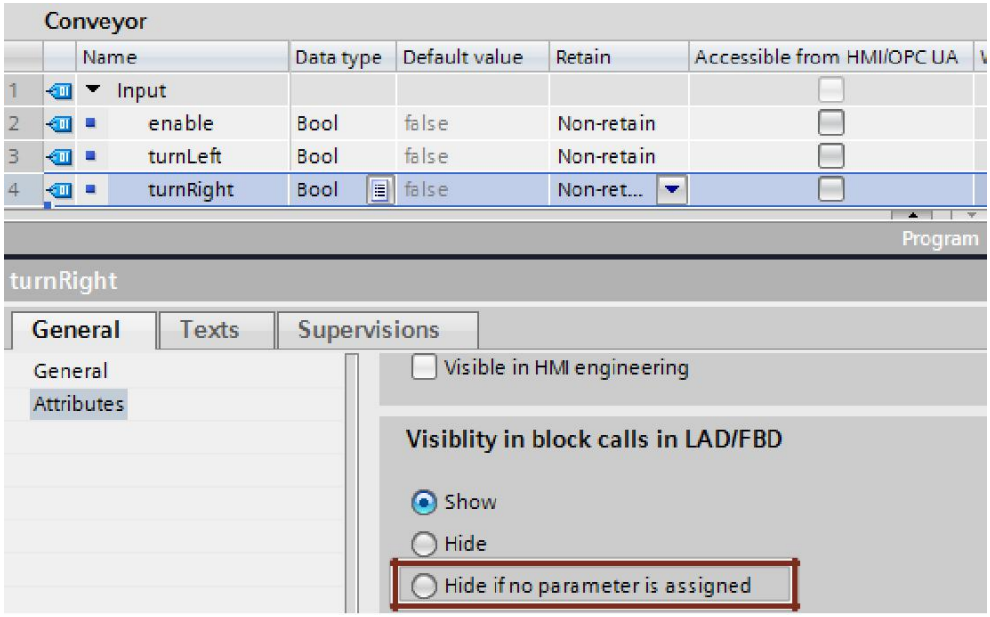

图 5-5

NF014 规则：有意义地格式化 SCL 代码

建议使用 TIA 博途的自动格式化功能。 这样做的好处是，所有用户都使用相同的格式， 同时缩进是自动完成的。

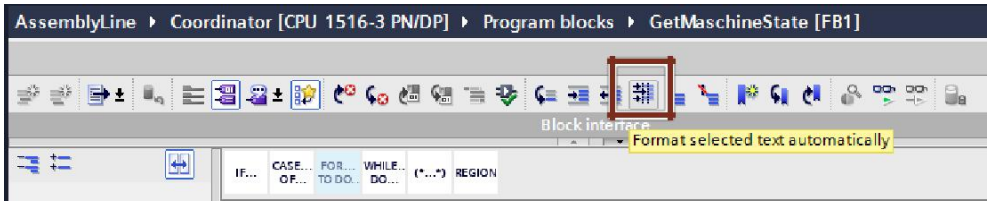

图 5-6

仅使用行注释“//”

有两种不同类型的注释:

- 块注释“(*…*)”或多行注释“(/*…*/)”， 用于描述一个函数或一个代码片段
- 行注释“//”描述单行代码，位于代码行的末尾或前面

若为了便于调试的目的进行代码屏蔽，则此时只允许使用行注释“//”。

注释的目的是向读者提供信息，说明代码意图。 注释的内容为写这段代码的原因。特别注意，注释不能是代码的冗余堆积，同时也不应该描述代码的功能细节（因为代码本身已经对此进行了描述）。相反，应描述编写代码的原因。

注意： 可以使用“注释”按钮，避免手动输入注释符号。

通过系统菜单支持此功能，以注释文本或删除注释符号。 此外，使用“(*...*)”或“(/*...*/)”，可以避免嵌套注释块的语法问题。

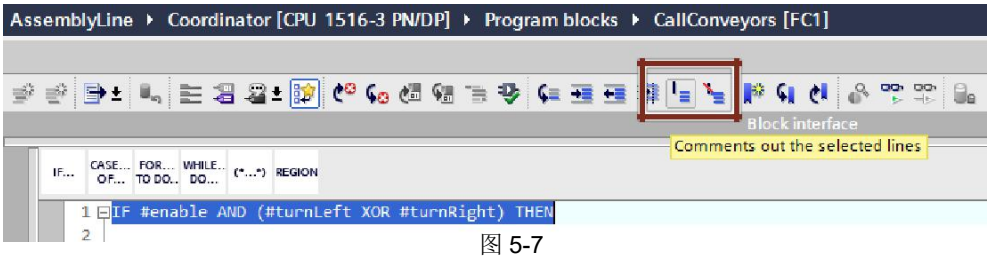

图 5-7

运算符前后的空格

在运算符的前面和后面必须使用空格。

表达式总是放在括号中

为了使解释顺序清晰，表达式根据所需的解释顺序放入括号中。

示例:

```
#tempSetFlag := (#tempPositionAct < #MIN_POS)
                    OR (#tempPositionAct > #MAX_POS);
```

条件和指令用换行符分隔

必须在条件和指令之间建立明确的区分。 这意味着，在一个条件（例如 THEN）之后或在一个替代分支（例如 ELSE）之后，必须在编程指令之前使用换行符。 此规则适用于其他类似构造的条件（例如，CASE，FOR，WHILE，REPEAT）。

示例:

```
IF #isConnected THEN // Comment
   ; // Statement section IF
ELSE
  ; // Statement section ELSE
END_IF;
```

部分条件下的换行符

在复杂表达式的情况下，用换行符突出显示每个部分条件是有意义的。运算符放在每一行部分条件表达式前面。如，将运算符放在新的条件前面。

示例:

```
#tempResult:=(#enable AND #tempEnableOld)
                OR (#enableAND #isValid
                  AND NOT (#hasErrorOR #hasWarning)
                );
```

条件和说明的适当缩进

控制结构体中的每个指令都必须缩进， 如果单行过长，布尔表达式将在下一行继续。

IF 语句中的多行条件缩进两个空格，THEN 新的一行中与 IF 是相同的缩进级别。 当 IF 条件适用于单行时，可以将 THEN 放在同一行的末尾。 在嵌套深度较深的情况下，THEN 指令单独放一行。 单个右括号指示嵌套条件的结束， 操作符总是在行的开头。

这些规则适用于其他类似控制结构的条件（例如，CASE，FOR，WHILE，REPEAT）。

示例:

```
IF #enable // Comment
  AND (
  (#turnLeft XOR #turnRight)
    OR (#statIsMaintenance AND #statIsManualMode)
  ) // Comment
  AND #tempIsConnected
THEN
  ; // Statement
ELSE
  ; // Statement
END_IF;
```

```
IF #enable THEN
  ; // Statement
  IF #tempIsReleased THEN
    ; // Statement
  END_IF;
ELSE
  ; // Statement
END_IF;
```

6 可重用性

本章描述了为确保程序元素的重复使用而建立的规则和建议。

RU001 规则：提供可以仿真的块

通过项目属性激活仿真功能。

说明: 通过此设置，程序块可以用在仿真环境中。

注意： 请注意在仿真环境中的专有技术保护会受影响。

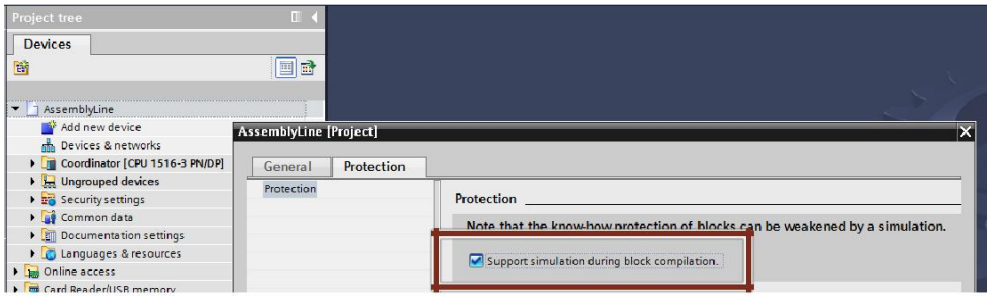

图 6-1

RU002 规则：完全使用库进行版本控制

版本控制必须完全执行。 这意味着必须记录对块的任何更改，并且必须维护版本号。 每次更改版本时，都会在适当的位置进行调整，例如，在 LAD 块的描述标题中。
使用库和块类型时，块版本由 TIA 博途集中管理。 在这种情况下，无需在块属性中手动维护版本。 更改历史记录不受影响。

将库升级到最新的 TIA 博途版本不会更改程序块，因此不影响库的版本。

注意： 在将块插入库之前，所有必要的设置，如自动进行编号，专有技术保护，仿真功能（通过项目属性）。 一旦块成为库的一部分，上面提到的设置就很难再更改。

在软件单元中的“发布”属性可在以后进行调整无需要修改类型。

版本号说明及使用

第一个发布的版本总是从 1.0.0 开始（参见表 6-1）。
第一个数字为最左边的数字。
软件版本号中的第三个数字的更改对功能或文档不受影响，例如错误修复。

若对库中的功能进行扩展，第二个数字将递增，第三个数字将重置。

对于新重大版本，如包含新功能或对以前版本不兼容的更改，增加第一个数字并重置第二个和第三个数字。

每个数字的缺省值范围在 0 到 999 之间。

表 6-1

Library	FB1	FB2	FC1	FC2	注释
1.0.0	1.0.0	1.0.0	1.0.0		新发布版本
1.0.1	1.0.1	1.0.0	1.0.0		FB1 中的错误修复
1.0.2	1.0.1	1.0.1	1.0.0		FB2 的优化
1.1.0	1.1.0	1.0.1	1.0.0		FB1 功能扩展
1.2.0	1.2.0	1.0.1	1.0.0		FB1 功能扩展
2.0.0	2.0.0	1.0.1	2.0.0		FB1 和 FC1 新的功能和可能不兼容的功能
2.0.1	2.0.0	1.0.2	2.0.0		FB2 中的错误修复
2.1.0/3.0.0	2.0.0	1.0.2	2.0.0	1.0.0	FC2 中的新功能/重大更新

FC，FB 和 PLC 数据类型的类型概念的使用

用户不必或可能不会调整的可重用函数、函数块和 PLC 数据类型作为类型存储在库中。

说明:只有这样，才能充分受益于 TIA 博途的类型概念。

注意： 此规则仅包含有关版本控制的一般信息。 关于库元素的自动版本控制的详细说明在库指南中提供:

https://support.industry.siemens.com/cs/ww/en/view/109747503

RU003 规则：在已发布的项目中只保留已发布的类型

已完成项目的库状态应是“已发布”，不能包含任何处于“测试中”的状态：

- 块（仅函数和函数块）
- PLC 数据类型

RU004 规则：只使用局部变量

在可重用块中，只能使用局部变量，不允许从 FC 或 FB 内访问全局数据。 全局数据可以通过块接口的形参传入。

将全局数据传递到 FC 可以用于:

- 访问全局 DB 和使用单实例 DB
- 全局计时器和计数器的使用
- 全局常量的使用
- 访问 PLC 变量

当满足上述要求时，TIA 博途会在块标题中以状态“对象满足库一致性”自动表示这一点。 因此，可以很容易地使用此状态来验证是否符合此规则。

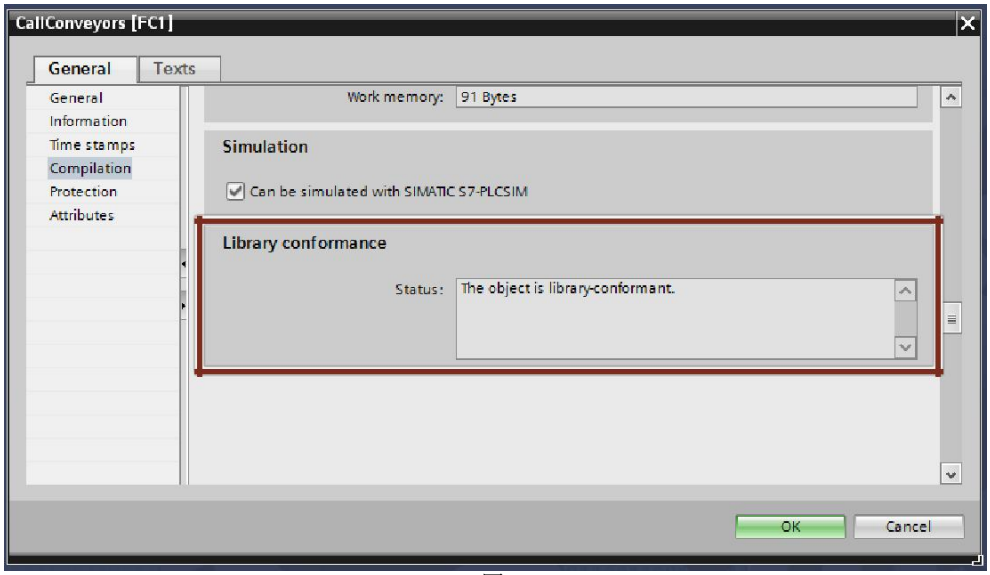

图 6-2

RU005 规则：使用本地符号常量

为了进一步封装块，可以使用局部常量。 当需要使用全局常量时，必须通过块接口的形参将它们传递到块中。 全局常量应在其自身的 PLC 变量表中定义。

注意: 当在块中使用全局常量时，更改其值需要重新编译该块。 对于专有技术保护的块，需要知道设置的密码。

禁止出现“魔数”

当代码中的变量做比较，或被赋值不是 0（整数）、0.0（实数/长实数）、TRUE 或 FALSE 时，应为此数值使用符号常量。

说明：通过这种方式调整该数值要容易得多，因为它集中在块接口中，而不是代码中的多个位置。

注意： 常量是数值的文本替换，由预处理器替换。 因此对 PLC 的性能或内存消耗没有负面影响。

可以将不同的符号常量分配给相同的值（包括等于 0 的值）。

示例:

```
STATUS_DONE                 WORD        16#0000
STANDSTILL_SPEED            LREAL       0.0
FREEZING_TEMPERATURE        LREAL       0.0
```

此外，由于符号标识符比数字更容易理解，所以可读性更强。

示例:

Blower

	Name	Data type	Default value	Retain
1	Input			
2	velocity	Real	0.0	Non-retain
3	Output			
4	InOut			
5	Static			
6	statVelocity	Real	0.0	Non-retain
7	Temp			
8	Constant			
9	MAX_VELOCITY	Real	10.0	

图 6-3

```
IF(#velocity < #MAXMAX_VELOCITY) THEN
  #statVelocity := #velocity;
ELSE
  #statVelocity := #MAX_VELOCITY;
END_IF;
```

RU006 规则：程序完全符号化

编程是完全符号化的，即程序中没有使用 ANY 指针等物理地址。

说明:由于符号化，这增加了程序的可读性和可维护性。

注意： ANY 指针的替代方法是 VARIANT 数据类型，也可以是 REF_TO 引用。 数据类型 VARIANT 可以提前检测类型错误并提供符号寻址。

RU007 建议：独立于硬件编程

为了保证不同系统之间的兼容性，建议仅使用与硬件无关的数据类型。

全局变量及系统变量的使用不能支持可重用性及硬件独立性。

这包括系统提供的定时器和计数器，如 S5-Timer。 鼓励使用符合 IEC 标准的类型，例如 IEC_Timer，它也可以用于多重实例中。

整个用户程序所需要的数据必须存储在全局数据块中。

注意： 硬件无关的系统函数对照表 可在西门子工业在线支持中获得，具体链接如下:

SIMATIC S7-1200/S7-1500 程序设计语言对照表

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

RU008 建议：使用模板

使用模板可以是所有程序员实现统一的基础。 模板提供的函数可以被认为是经过验证的，并且可以显著地减少开发时间。

另一个优势是，由于块提供相同的基本功能，统一的块使得更容易使用及改进。例如，参考 PLCopen 标准。

注意： 所引用的模板可以在通用函数库（LGF）中作为主副本找到：

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

7 引用对象（分配）

本章介绍了内存管理和访问的规则和建议。

AL001 规则：使用多重实例而不是单个实例

在程序中，最好使用多重实例而不是单个实例。

说明: 使用这种方法，可以将函数块做成封装的形式，在上层结构或全局结构中不需要额外的实例，从而减少了对象的数量。

AL002 建议：定义从 0 到常数值的数组边界

数组边界从 0 开始，以一个符号常量作为数组的上边界结束。

- 对于块内部的数组，必须在块接口的本地数据中定义常量
- 对于全局数据块和 PLC 数据类型中的数组，用作上限的常数必须在 PLC 变量表中定义
- 常量以及用于访问数组元素的索引的数据类型，从性能方面的原因应使用 DINT

示例:

```
BUFFER_UPPER_LIMIT     DINT    10


diagnostics            Array[0.. BUFFER_UPPER_LIMIT] of typeDiagnostics
```

说明: 数组索引以 0 开头有几个好处：（1）某些系统指令和数学运算是基于零开始工作的，例如模函数，通过这种方式，索引可以直接用于此类函数而无需进行任何调整。（2）WinCC（Comfort，Advanced，Professional and Unified）可以处理脚本中基于零的数组。

在数组边界不能以零为基础的情况下，则应将符号常量用于上限和下限。

AL003 建议：将数组参数声明为数组[*]

如果必须将数组作为形参传入，建议将其作为未指定大小的数组传入。

可通过系统函数“UPPER_BOUND 和“LOWER_BOUND”可以确定大小和限值。

示例:

```
diagnostics            Array[*] of typeDiagnostics
```

说明: 这样做可以创建泛型程序结构。 特别是在专有技术保护的块中，由于块接口中没有显式声明大小，因此不需要重新编译。

AL004 建议：指定所需的字符串长度

“String”和“WString”总是保留存储 254 个字符所需的内存。 一个“String”最多可以包含 254 个字符，一个“WString”最多可以包含 16382 个字符。 建议使用符号常量将字符串限制在所需要的长度。

说明: 此过程可防止系统分配过多的内存。 此外，当传入每个形参赋值的字符串时，它还提供了性能优势。

示例:

```
MAX_MESSAGE_LENGTH      DINT     24

errorMessage            String[#MAX_MESSAGE_LENGTH]
```

8 安全

本章介绍了创建尽可能强壮且安全的程序所需要的规则和建议。

SE001 规则：验证实际值

应验证传入的所有实际值，以避免不受控制和意外的程序执行和状态。

如果实际值不可信或无效，则必须向用户提供指示，请参见“DA013 规则:通过“status”/“error”报告状态/错误。

SE002 规则：初始化临时变量

块中使用的每个临时变量都必须在首次使用之前进行初始化，直接赋值，运算输出赋值，或使用数据类型中的常量赋值。

另请参阅“NF012 规则：以对应的格式初始化变量“

说明: 由于系统只初始化基本数据类型，剩余其他未定义的值可能导致意外的程序行为。

注意： 使用带有工艺块的变量，小于 0.0 的值具有特殊的含义，根据形参而不是变量值，将使用工艺对象中预先配置的默认值。

因此，应选择适当的初始值。

SE003 规则：处理 ENO

在启动输出 ENO 的帮助下，可以检测运行时错误。
以下指令的执行取决于 ENO 的信号状态。

说明: EN/ENO 机制的使用避免了意外程序的中断。 块状态以 Bool 变量的格式传递。

为了提高 PLC 的执行性能，可以取消自动 EN/ENO 机制。 这样做的结果是，不能再使用 ENO 值来响应运行时错误，激活以下指令必需手动实现。

因此在任何情况下，对 ENO 的赋值都应出现在程序中。 最简单的形式是:

```
ENO := TRUE;
```

SE004 规则：有选择地激活 HMI/OPC UA/Web API 数据访问

默认情况下，应禁用对每个变量的 HMI/OPC UA/Web API 访问，不允许访问 FB 的静态变量，必须创建用于读或写访问的变量。

通常，通过 HMI/OPC UA/Web API 的可访问性必须至少在 PLC 数据类型编辑器中激活，以允许访问。 当使用 PLC 数据类型时，必须在块接口中相应地调整对它的访问。

SE005 规则：评估错误代码

如果使用的 FC，FB 或系统函数向程序提供错误代码错误，则必须对它们进行评估。

用户程序无法处理的错误，则必须向用户提供明确的错误消息，以便定位错误原因。

注意： 有关“错误处理”主题的进一步信息，请参阅“DA013 规则：通过“status”/“error”返回状态/错误”。

SE006 规则：用评估逻辑写错误 OB

如果 OB 组织块用于处理错误，那么它们将完成特定的任务。

最低要求是在用户程序中评估错误，并报告和处理错误。

9 设计指南/体系架构

本章描述程序设计和程序体系架构的适用规则和建议。

DA001 规则：对项目/库进行结构化和分组

将程序拆分成逻辑单元，系统提供了几种不同的方法：

- 将相关的程序块合并到一个文件夹/组中
- 将工艺机械零件组成软件单元
- 将程序结构化为逻辑功能单元-FC/FB
- 将相关的数据整理成 PLC 数据类型
- 通过网络（NETWORK）或区域（REGEON）结构化程序

注意： SCL 中的区域(REGION)相当于 LAD/FBD 中的程序段(NETWORK)。

区域(region)名称和网络(network)标题类似，应同样编写。

区域提供了几个好处:

- 所有区域的概述在编辑器的左边
- 在概述的帮助下快速浏览代码，并在代码内部链接
- 代码片段可折叠
- 通过概览和代码同步，借助导航快速折叠和展开

DA002 建议：使用适当的编程语言

使用适合编程任务的编程语言。

标准块-结构化文本（SCL/ST）

标准块的首选编程语言是 SCL。 SCL 提供了所有编程语言中最紧凑的形式和最好的可读性，支持程序员的选定某一代码元素，该元素在同一程序块中出现了的所有位置自动高亮显示。

调用环境-图形化/面向块（LAD，FBD）

如果几个块应该互连，例如在 OB 调用环境中，那么编程语言 LAD 或 FBD 是最合适的。 此外，在块中包含大部分二进制逻辑判断的情况下，应使用 LAD 或 FBD，可以更容易的诊断，并为提供更快的预览。

顺序控制-面向流(GRAPH)

当涉及顺序控制编程时，**GRAPH** 是首选语言。 使用 GRAPH，可以快速地对顺序步编程，并且可以很容易地跟踪执行。 此外，该系统已经提供了“互锁”和“监控”的功能。

DA003 规则：设置/评估块属性

必须在块属性中激活以下设置:

- **自动编号:** 块（OB，FC，FB，DB，TO）应该在发布之前启用自动分配编号。请注意，组织块的执行取决于其序号/优先级
- **IEC 检查:** 为确保符合 IEC 编程，IEC 检查必须 打开。 只有这样才能确保类型兼容和类型安全编程
- **优化访问:** 若想要完全符号化编程得到最大性能必须激活对块的优化访问

在块属性中，应检查以下属性:

- **ENO:** 请参阅 “SE003 规则：处理 ENO”。
- **多重实例功能:** 如果块内部使用多重实例而不是全局单实例，勾选多重实例能力该块才能正常运行。
- **库一致性:** 请参阅 “RU004 规则:只使用局部变量”

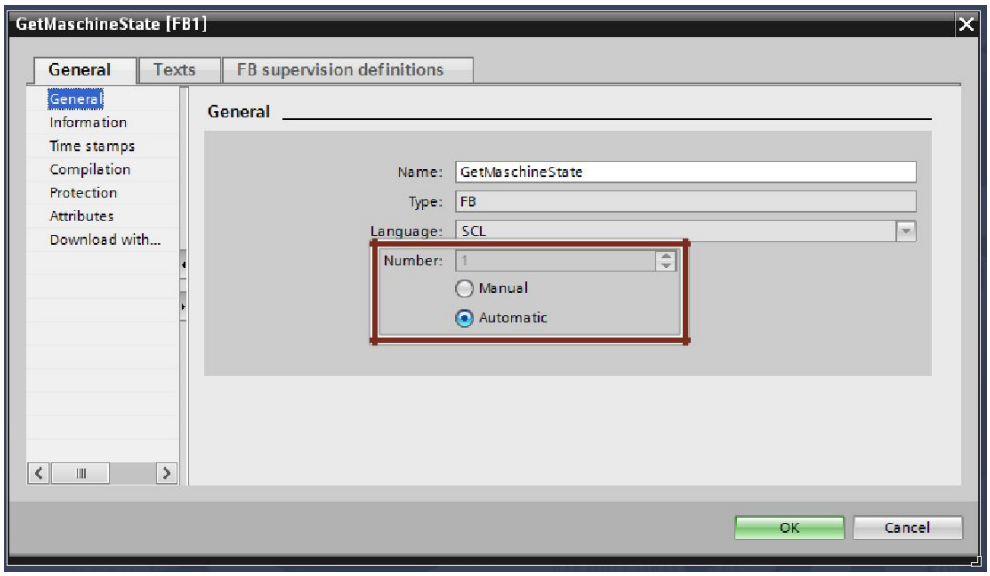

图 9-1：自动编号

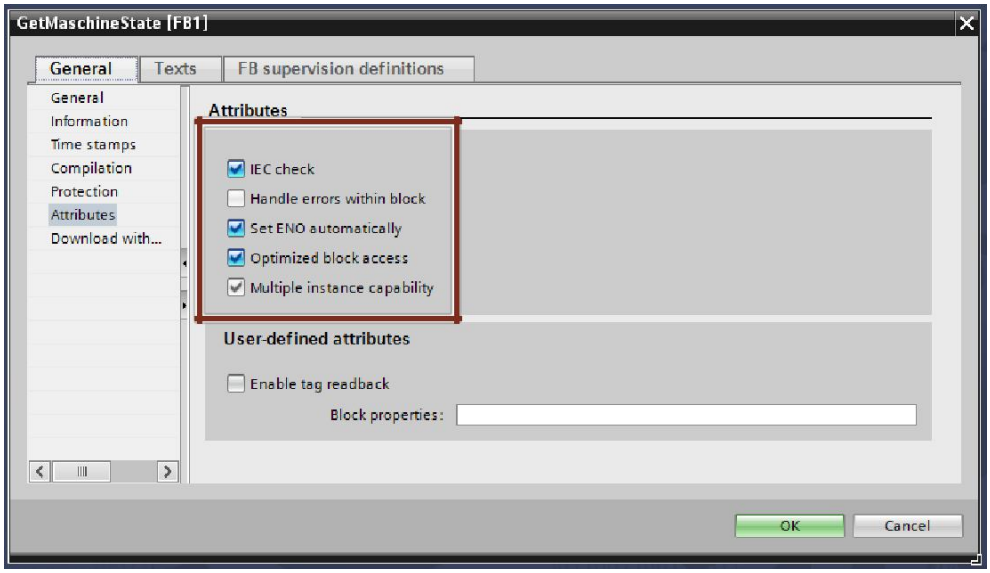

图 9-2：IEC 检查，ENO，优化访问，多重实例可用性

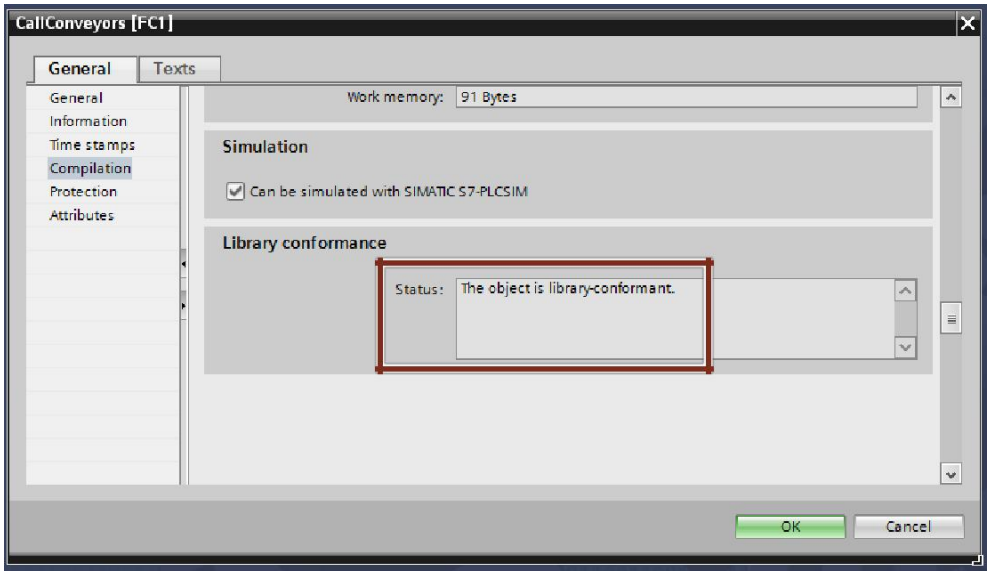

图 9-3：库一致性

DA004 规则：使用 PLC 数据类型

PLC 数据类型应用于用户程序的结构化。 在本地数据中，当变量作为一个整体传送时，也使用 PLC 数据类型。

结构体（STRUCT）只在块的本地数据中声明，以更容易理解的方式对变量进行分组，但不能用于数据交换。

说明: PLC 数据类型的更改将在所有位置自动更新，通过形参简化了多个块之间的数据交换。

DA005 规则：只通过形参交换数据

如果多个 FB 或 FC 中需要数据交换，则只能通过 Input，Output 或 InOut 参数进行数据交换。

说明：在封装块的意义上，数据与这些可重用的块断开连接，这样解决了可重用块的依赖关系。 由于数据是通过形式参数传入的，因此不需要修改该块。 调用者仍然可以控制哪些数据在何处使用。保证了多次访问（可能在程序内的不同地方）情况下的数据一致性。

DA006 规则：仅从块内访问静态变量

函数块的静态数据只能在声明它们的块内使用。

说明: 通过直接访问实例的静态变量，兼容性无法保证，因为将来的更新会影响到相应的访问。 此外，静态变量的修改对 FB 执行的影响无法判断。

DA007 建议：形参组

当有许多（例如超过十个）参数要传递时，这些参数应被分组到一个 PLC 数据类型中。 此参数应声明为 InOut 参数，并将作为“引用调用”传递。

此类参数的示例是配置数据，实际值，设定点或函数块的诊断数据的输出。

请参阅“PE003 建议：使用引用传递结构化参数"

注意： 在控制变量和状态变量经常变化的情况下，在 LAD/FBD 中直接使用这些变量，并将它们声明为基本输入或输出参数可以方便监控。

DA008 规则：输出参数只写一次

每个执行周期只写入一次输出变量和返回值。 这应在可能的情况下，在函数块的末端集中进行。

不允许读取自己的输出参数或返回值，必须使用临时或静态变量。

说明: 这样可以确保所有输出值的一致性。

DA009 规则：仅保留使用过的代码

在发布的程序中，只应包含正在 PLC 中执行的代码。

违规示例:

- 从未调用过的块或工艺对象
- 从未使用的变量或常量
- 从未使用的参数
- 从未执行的程序代码
- 注释掉的代码
- 从未访问过的 PLC 变量
- 从未使用过的用户常量
- 外部源文件

注意： 在不同时间点所需的生产代码具体取决于选项,不受此影响。

DA010 规则：根据 PLCopen 开发异步块

PLCopen 组织已经定义了运动控制块的标准。 这个标准可以应用于所有异步块。从这个角度讲，在多个周期内处理的所有块都遵循此标准，例如用于通信，闭环控制或运动控制块。

说明: 应用此标准可以简化编程和实现应用。

DA011 规则：带“enable”的连续异步执行

仅启动和初始化一次，之后仍保持运行以响应输入的块具有“enable”输入参数。

示例: 通信块（充当服务器）在初始化后等待来自客户端的连接请求。 在成功的数据交换之后，服务器等待其他的连接请求。

注意： 可以在通用函数库(LGF)的主副本中找到带 enable 的模板：

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

设置参数“enable”将启动异步任务的执行。 如果“enable”保持设置，则任务执行将保持活动状态，并且正在接受和处理新的值。

当“enable”参数被复位时，作业结束。

诊断信息（诊断）将在“enable”上出现新的上升沿时复位。

如果根据 PLCopen 实现该块并使用“enable”输入参数，则必须至少提供输出参数“valid”，“error”和“busy”。

表 9-1

标识符	数据类型	描述
输入参数		
enable	Bool	所有参数由输入的“enable”上升沿激活 且随时可以被不断的修改。 为电平触发激活（TRUE）和禁用（FALSE）。
输出参数		
valid	Bool	输出“valid”和“error”为**互斥的；** 如果输入使能且输出值是有效的， 则参数“valid”输出，一旦检测到“error” 则输出“valid”复位。
busy	Bool 参数“BUSY” 设置为输出	FB 正在执行命令。 新输出值计算中。 只要 FB 执行命令，参数“BUSY”设置为输出。 输出“busy”会在 enable 的上升沿置位，并在 FB 命令执行过程中保持置位状态。
error	Bool	输出“valid”和“error”为 **互斥的**。 输出的上升沿表示在执行 FB 期间发生错误。
commandAborted	Bool	**可选**输出,它表示当前执行的 FB 任务被 另一功能或同一对象的另一任务取消。 示例：一个轴正在定位， 而另一个函数块在执行停止该轴。 这时定位函数块将输出“commandAborted”这个参数， 表示定位命令被停止命令中止。
status	Word	**可选**输出：函数块中的错误和状态信息， 此参数的命名来自系统函数 （根据“PLCopen 标准为“errorID”）。
diagnostics	PLC data type	**可选**输出：详细的错误信息。 在这里，所有的错误信息和警告都会储存。 诊断信息的结构在建议“传递底层的状态信息” 中进行了描述。

示例：

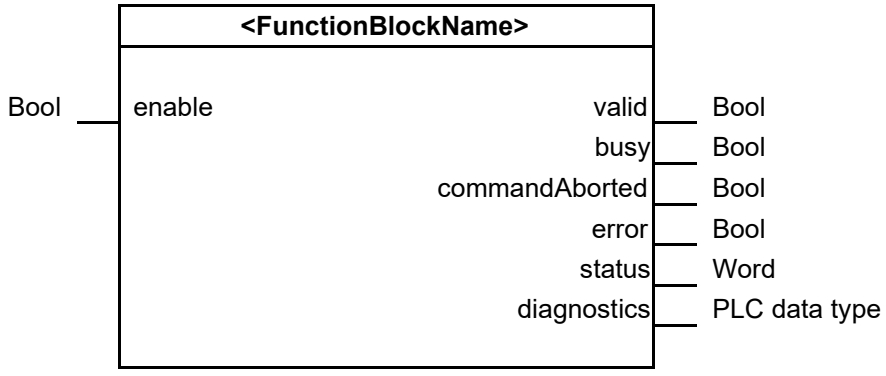

图 9-4

带“enable”程序块的信号图

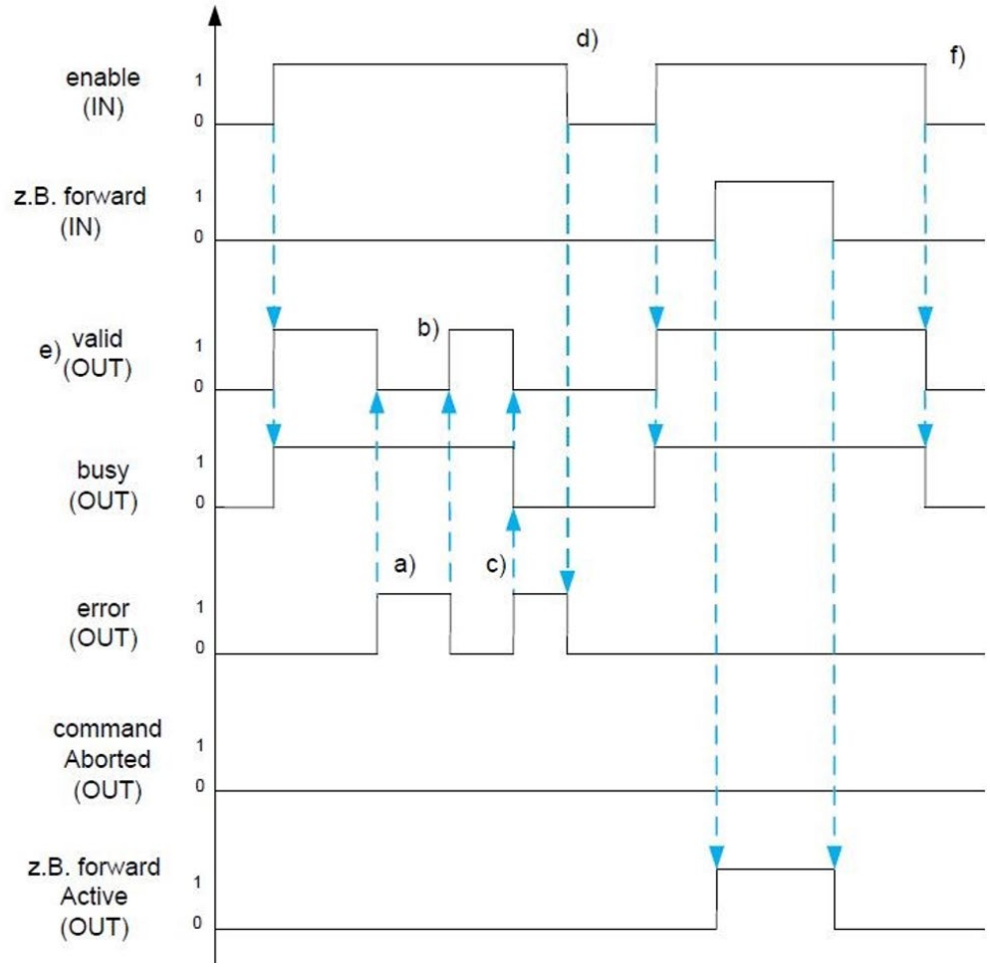

图 9-5

a) 当“error”为 TRUE 时，将复位“valid”，并停止块内的所有功能。 由于错误可以由块本身处理，所以“busy”标志保持活动状态。

b) 清除错误原因（例如重新建立连接）后，“valid”再次变为激活状态。

c) 发生只能由用户清除的错误时，必须置位“error”，并复位“valid”和“busy”。

d) 必须由用户纠正的未决错误只能通过“enable”处的下降沿进行确认。

e) “valid”为 TRUE 表示块已激活，没有错误并且 FB 的输出有效。

f) 如果“enable”复位为 FALSE，则“valid”和“busy”也必须复位。

只要信号“enable”被置位，“commandAborted”，“error”和“done”就必须被置位至少一个执行周期。

DA012 规则：带“execute”的单次异步执行

一次处理的块具有输入参数“execute”。

示例: 一个通信块（客户端）只请求一次服务器的数据。 这是由输入信号“execute”处的边沿触发的。 在响应被处理之后，处理结束。若需重新触发需要“execute”的下一个边沿信号。

注意： 带有“execute”的块模板可以在通用函数库(LGF)中的主副本中找到:

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

“execute”的上升沿启动任务，输入参数处的值被应用。

任务开始后对值所做的任何更改只有在新任务开始后才会生效，除非正在使用“continuousUpdate”。

参数“execute”的复位不会停止当前任务的执行，但对运行状态的显示持续时间有影响。 如果“execute”在当前任务完成之前被复位，那么参数“done”，“error”和“commandAborted”将只被置位一个周期。

诊断信息（“diagnostics”）只有在“execute”出现新的上升沿时才会被清除。

完成任务后，“execute”的新上升沿是启动新任务所必需的。这可确保每次启动任务时，块都处于其初始状态，并且该函数的处理独立于先前的任务。

如果根据 PLCopen 标准实现该块，并且使用“execute”输入参数，那么必须使用输出参数“busy”，“done”和“error”。

表 9-2

标识符	数据类型	描述
输入参数		
execute	Bool	**无“continuousUpdate”参数的“execute”:** 所有参数均在“execute”上升沿后执行且函数块启动。当输入参数更改时，需要一个新的“execute”上升沿才可生效。 **有“continuousUpdate”参数的“execute”:** 所有参数均在“execute”上升沿后执行，如果“continuousUpdate”为 TRUE 时参数值调整即生效。
continuousUpdate	Bool	**可选** 输入: 参见“execute”输入。
输出参数		
done	Bool	输出“done”，“busy”，“commandAborted”和“error”是 **互斥的**。 如果命令执行成功则参数“Done”输出。
busy	Bool	输出“done”，“busy”，“commandAborted”和“error”是 **互斥的**。 FB命令未执行完成，意味着会有新的输出值。 “execute”的上升沿时，“busy”置位；当输出“done”、“commandAborted”或“error”之一置位时，“busy”复位。

标识符	数据类型	描述
error	Bool	输出“done”，“busy”，“commandAborted”和“error”是 **互斥的**。 输出的上升沿表示错误在 FB 的执行期间发生。
commandAborted	Bool	输出“done”，“busy”，“commandAborted”和“error”是 **互斥的**。 **可选**输出，指示当前执行的命令被同目标的其它功能或其它命令终止. 示例：一个轴正在定位，而另一个函数块在执行停止该轴，这时定位函数块将会输出“commandAborted”这个参数，表示定位命令被停止命令中止。
status	Word	**可选** 输出：函数块中错误和状态信息 该参数是根据现有的系统函数命名的。（根据 PLCopen 为“errorID”）
diagnostics	PLC data type	**可选** 输出:详细的 error 信息。 在这里，所有的错误信息和警告都会储存。 诊断信息的结构在建议“传递底层的状态信息”中进行了描述。

示例:

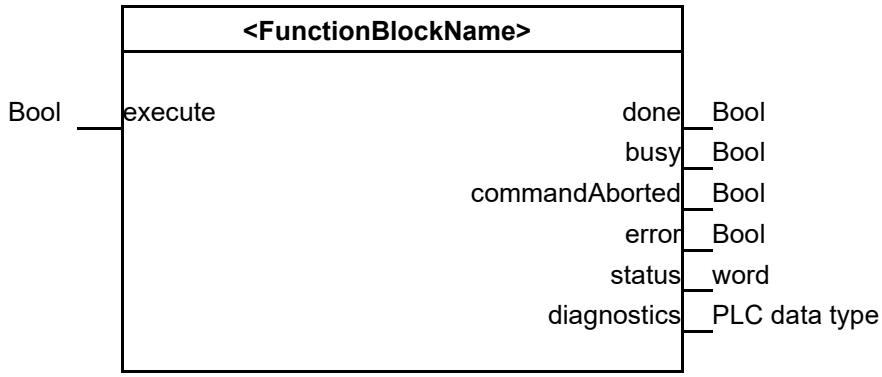

图 9-6

带有“execute”的块的信号图:

注意： **如果在置位输出参数“done”或“error”之前复位输入参数“execute”，则“done”或“error”只能置位一个周期**

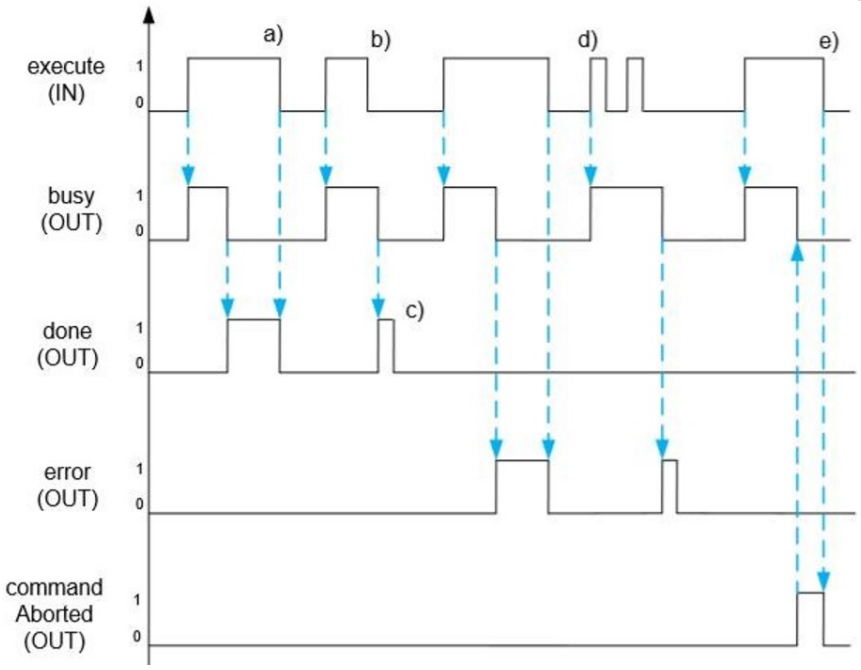

图 9-7

a) “done”，“error”和“commandAborted”将在“execute”的下降沿复位。

b) FB 的功能不会因“execute”的下降沿终止。

c) 如果“execute”已为 FALSE，则“done”，“error”和“commandAborted”仅置位一个周期。

d) 当前一个命令仍在执行（“busy”=TRUE）时，一个的新的命令请求被“execute”的上升沿发起，上一个命令应使用以前使用的参数完成，或者上一个命令应中止并使用新的参数重新启动。具体的执行应参考不同的应用场景来判断且必须详细说明记录。

e) 如果命令的执行被具有相同或更高优先级的另一个命令（来自另一个块/实例）中断，则该块参数“commandAborted”输出且立即停止执行的任何剩余命令。 当发出紧急停止，同时轴执行定位命令时，可能会发生这种情况。

DA013 规则：通过“status”/“error”返回状态/错误

块在其输出参数“status”处报告唯一的状态，该参数提供有关块内部状态的信息。这些值必须定义为块接口中的局部符号常量，以避免重复使用并增加可读性。

块发生错误时，应使用输出参数“status”和“error”。 按照下面状态概念的描述，“error”参数是“status”的最高位（位 15）。 其余位用于指示错误代码，从而可以识别错误的原因。

出于与 SIMATIC 系统块兼容的原因，输出“errorID”（在 PLCopen 标准中是强制性的）被输出“status”替换。

说明：这允许通过输出“status”传递关于块状态的进一步详细信息，其中可以不包含任何错误信息。

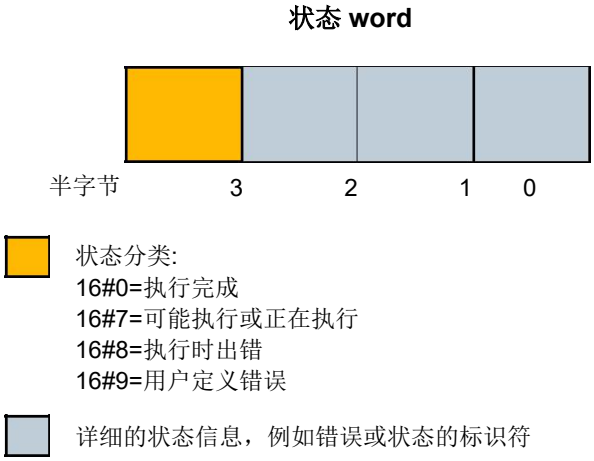

图 9-8

DA014 规则：“status” 使用标准化的取值范围

对于输出参数“Status”的标准化，应使用以下定义的信息和错误值范围。

表 9-3

信息	值范围
命令完成，没有警告，也没有进一步的详细信息	16#0000
命令完成，详细信息	16#0001...16#0FFF
没有正在执行的命令（初始值）	16#7000
收到新命令后的第一次调用 （上升沿开启执行）	16#7001
在命令激活后续调用中执行期间，无详细信息提供	16#7002
在命令激活后续调用中执行期间，有详细信息提供 发生的警告不会进一步影响操作	16#7003...16#7FFF

表 9-4

错误	值范围
函数块操作错误	16#8001...16#81FF
参数设定错误	16#8200...16#83FF
从外部执行时出错 （例如错误的 I/O 信号，轴未归位）	16#8400...16#85FF
执行过程中出现内部错误 （例如在系统调用期间）	16#8600...16#87FF
保留	16#8800...16#8FFF
用户定义的错误类	16#9000...16#FFFF

DA015 建议：传递下层信息

如果一个块调用其他子函数，这些子函数报告详细的状态和可能的诊断信息，那么它们必须复制到输出参数“diagnostics”处的诊断结构中。 此外，该诊断结构可以包含用于诊断附加值，例如运行时信息。

注意： 诊断结构可能进行掉电保持，即使在电源故障之后也可以进行诊断。

简单诊断结构示例：

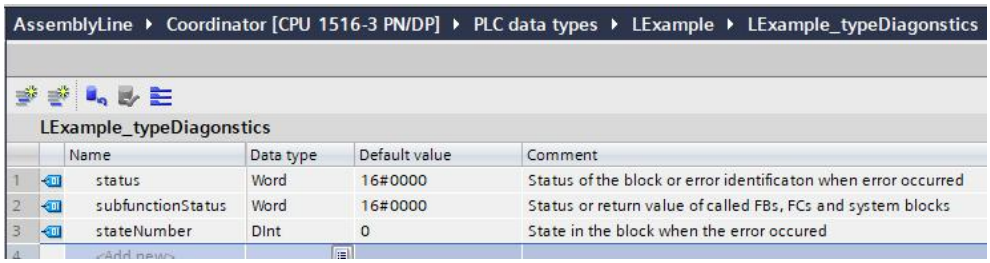

图 9-9

简单诊断结构包含三个参数:

表 9-5

标识符	数据类型	描述
status	Word	当前块的状态
subfunctionStatus	Word/DWord	下层子功能的状态
stateNumber	DInt	发生错误时，执行状态/执行步骤的编号

扩展诊断结构示例:

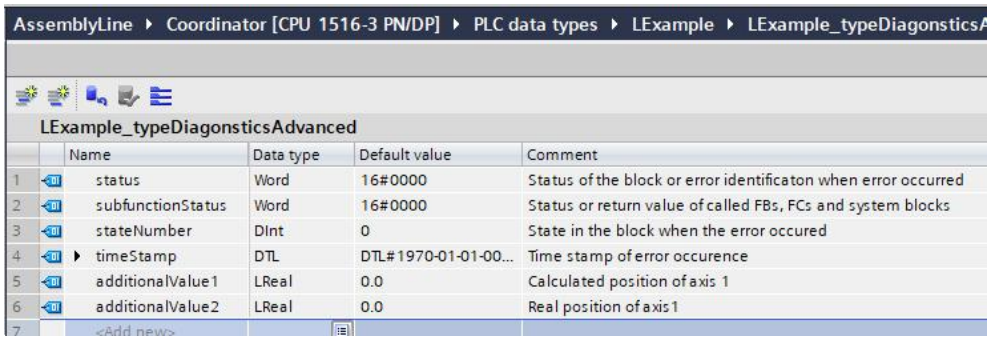

图 9-10

变量“timeStamp”包含错误发生的时间点。

在“stateNumber”中存储内部状态机的当前内部状态。

如果系统函数或被调用的 FB/FC 出现错误，其状态应存储在变量“subfunctionStatus”中。

输出参数的唯一错误代码应复制到诊断结构的变量“status”中。

可以使用适当的数据类型将修正错误的附加变量（也是基础变量）添加到诊断结构中，例如借助“additionalValueX”，其中“X”由从 1 开始的递增数代替。

DA016 建议：用 CASE 指令代替 ELSIF 分支

在可能的情况下，使用 CASE 指令而不是带有多个 ELSIF 分支的 IF 指令。

说明:程序变得更易读。

DA017 规则：在 CASE 指令中创建 ELSE 分支

CASE 指令必须始终具有 ELSE 分支。

说明:这起到报告错误的作用，这些错误可能在运行时发生。

示例:

```
CASE #stateSelect OF
  CMD_INIT:// Comment
    ; // Statement
  CMD_READ:// Comment
    ; // Statement
ELSE
  // default statement
  ; // Generate error message
END_CASE;
```

DA018 建议：避免跳转和标签

尽可能避免程序内的跳转。 只有在例外的情况下，如没有其他可能实现程序的方法，跳转才是允许的。

说明：跳转可能导致这些指令从程序内部的来回跳转，因此很难保证程序正常运行。

10 性能

在本章中描述了高性能用户程序开发的规则和建议。

PE001 建议：禁用“创建扩展状态信息”

禁用 “创建扩展状态信息”可以使生产运行达到更好的性能。 在开发过程中，为了调试用户程序，激活此设置可能是有益的。

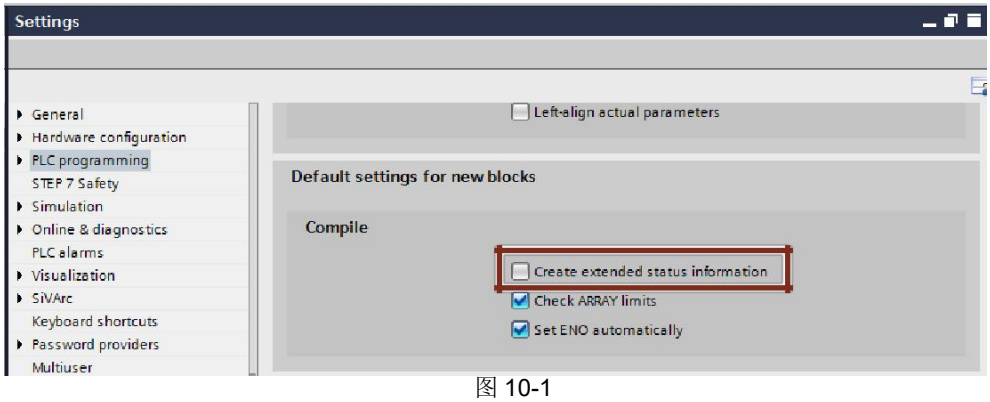

图 10-1

PE002 建议：避免“在 IDB 中设置”

为了允许块优化，并且为了完全符号编程，应避免在块接口中使用“在 IDB 中设置”的功能。

说明:使用“在 IDB 中设置”会导致系统创建一个由优化和非优化数据区域组成的混合 DB。 访问这些数据会导致系统将这些数据复制到其他数据格式中。

注意： “在 IDB 中设置”主要与“AT 结构”结合使用。可以使用片段访问或系统函数 SCATTER 和 GATHER 替代。

PE003 建议：使用引用传递结构化参数

为了保证最大性能（内存和运行最佳）的将数据传递到块接口的形参中，建议使用“按引用调用”模式。

说明:调用块时，将传递对实际参数的引用，不复制实际参数。

注意： 使用这种方法，可以修改原始数据。

在调用块时, 如果将优化的数据传递给禁用了“优化的块访问”属性的块（反之亦然），则该数据总是作为复制进行传递。 如果块包含许多结构化参数，这会导致块的临时内存区域溢出。 您可以通过为两个块设置“优化”访问类型来避免这种情况。

下表总结了如何在 SIMATIC S7-1200 / S7-1500 PLC 中使用基本或结构化数据类型传输形式参数。

表 10-1

块类型/形参		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

PE004 建议：避免 Variant 形参

为了避免由于使用 Variant 造成的性能损失，建议为不同的数据类型设置单独的块。
仅在以下情况下才需要使用 Variant。 例如，需要将用于通信的数据传输到块内部调用的通信系统函数，或需要做序列化。

PE005 建议：避免形参 “mode”

避免开发根据输入参数（例如“mode”）操作不同功能的块。

说明：这可以防止不需要的代码片段（“死代码”），因为模式参数通常是静态连接的。

相反，您应该将不同功能分到不同的程序块:

- 这减少了内存消耗，并通过减少代码来提高性能
- 它通过更好的区分和更好的命名来增加可读性
- 它通过更小的代码片段来增加可维护性，这些代码片段彼此独立

PE006 建议：首选临时变量

如果变量仅在当前周期中需要，则应将它们声明为临时变量。 临时变量在块中可提供最好的性能。

如果经常访问 Input 或 In/Out 参数，则应使用一个临时变量作为缓存，以改善执行时间。

注意： 不能在监视表或强制表中监视或强制临时变量。

PE007 建议：将重要的测试变量声明为静态

重要的测试变量应该静态声明，这些变量必须提供关于函数状态的足够信息。

说明：即使在块已完成执行之后，静态测试变量的当前值仍保留，以用于诊断目的。

PE008 建议：将控制/索引变量声明为“DInt”

对于循环，迭代和数组访问的控制和索引变量，建议使用数据类型“DInt”。

说明：这种数据类型具有最高的处理器处理性能，因为不必进行类型转换。 因此，数组大小和循环限值的定义也应创建为“DInt”数据类型的常量。

PE009 建议：避免多个相同的索引访问

为了避免重复访问数组的同一索引，应使用临时变量作为缓存。

说明:这种方法将数组边界的内部系统检查和超过边界的检查减少到最小。

示例:

```
For #tempIndex 0 to #MAX_ARRY_ELEMENTS DO
  // Copy to temporary variable
  #tempCurrentData:=#statArray[#tempIndex];
  //  Reset all member variables
  #tempCurrentData.element1:= FALSE;
  #TempCurrentData.element2:= FALSE;
  #TempCurrentData.element3:= FALSE;
  #TempCurrentData.element4:= FALSE;
  #TempCurrentData.element5:= FALSE;
  // Write back the changes made
  #StatArray[#tempIndex]:=#tempCurrentData;
END_FOR;
```

PE010 建议：使用片段访问代替掩码

应使用片段访问来访问单个位，而不是使用掩码来访问几个独立位。

说明:这种方法提高了源代码的性能和可读性。

示例: 计算 Bit1 = TRUE and Bit0 = FALSE 时使用片段访问

```
#tempIsTriggered:=(#trigger.%X1 AND NOT #trigger.%x0);
```

当要将单个位与完整变量进行比较时，建议采用掩码。

示例:计算 Bit1=TRUE and Bit0 = FALSE 时使用掩码

```
PATTERN_MASK        BYTE      2#00000011
PATTERN             BYTE      2#00000010

#tempIsTriggered:=((#trigger AND #PATTERN_MASK)=#PATTERN);
```

PE011 建议：简化 IF/ELSE 指令

将 IF/ELSE 指令简化为简单的二进制运算，可以提高性能，减少内存消耗。

边缘检测的错误示例:

```
// Check for rising edge
IF #trigger AND NOT #statTriggerOld THEN
 #tempIsTrigger := TRUE;
ELSE
 #tempIsTrigger := FALSE;
END_IF;
// Store trigger for next cycle
#statTriggerOld := #trigger;
```

正确示例:

```
// Check for rising edge
# tempIsTrigger:=#trigger AND NOT #statTriggerOld;
//  Store trigger for next cycle
#statTriggerOld:= # trigger;
```

PE012 建议：根据预期对 IF/ELSIF 分支进行排序

IF/ELSIF 语句，则应按低可能性的方式排序，最有可能的情况应排在前面，依此类推。

说明:这避免了对不太可能的情况的评估，从而提高了效率。

示例: 假设程序流无错误地执行，并且理想情况已知，则首先评估最可能的情况。

```
// Check if connection isestablished
IF #instConnect.done=TRUE THEN
// Connection is established – set next state
;
 // Check if TCON throws an error
ELSIF # instConnect.error=TRUE THEN
//TCON throws an error - do error handling
;
END_IF;
```

PE013 建议：避免内存密集型指令

会使存储空间占用率过高的指令（内存密集型指令），如:

- “GetSymbolName”
- “GetSymbolPath”
- “GetInstanceName”
- “GetInstancePath”

应尽量避免使用。

说明：使用上面提到的指令会导致存储器空间需求大幅增加， 指令的调用频率越高，符号标识符越长，对存储器的空间要求也就越大。

PE014 建议：避免运行时密集指令

应将运行时密集指令的使用减少到最低限度。

运行时密集指令是处理大量数据的指令，如“Serialize”和“Deserialize”或者访问存储卡的指令。

系统函数“WRITE_DBL”，“READ_DBL”，“FileRead”和“FireWrite” 访问相对较慢的 SIMATIC 存储卡，通常是异步完成的，可能需要几个循环。 但是，由于要传输大量数据，因此使用这些系统函数会对整个程序的运行时间产生负面影响。
访问 SIMATIC 存储卡的更好的方法是 DataLog 和 Recipe 函数。

PE015 建议：SCL/LAD/FBD 用于运行时间敏感的应用

对于时间要求严格的程序/程序部分和算法，建议使用三种编程语言 SCL，LAD 或 FBD 中的一种。

说明：GRAPH 作为编程语言生成额外的诊断信息，这些信息需要额外的运行时间。 推荐使用 GRAPH 对机器流程进行顺序编程。

PE016 建议：检查最小循环时间的设置

对于没有高通信负载的时间关键应用，可以关闭“最小循环时间”，以允许快速响应时间。

高通信负载可以通过启用并提高“最小循环时间”来实现更高的通信吞吐量。

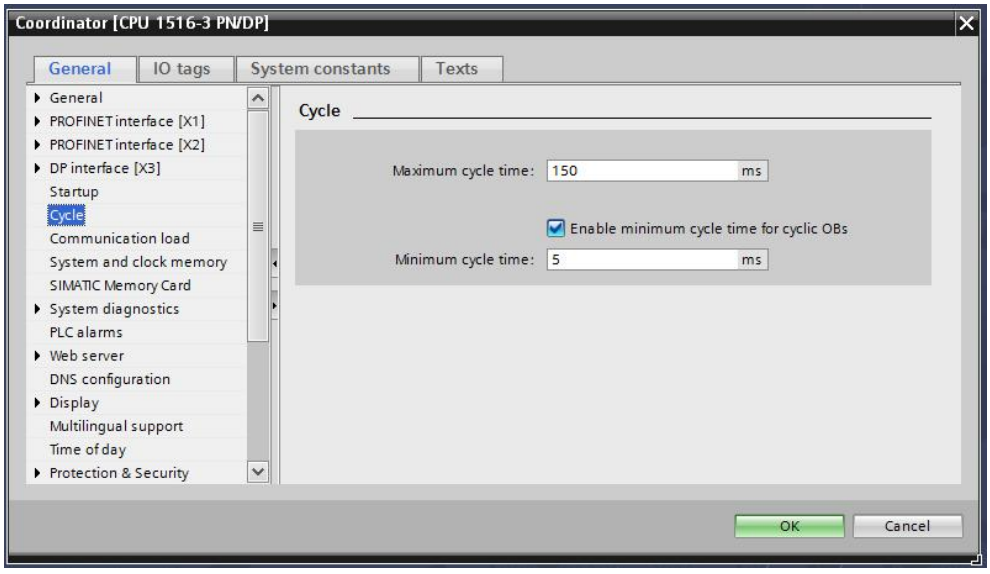

图 10-2

11 速查表

11 Cheat sheet

Block interface	*In*	*Out*	*InOut*	*Stat*	*Temp*	*Const*
	enable	**done**	**conveyorAxes** **instTimer**	**statState** **instTimer** **powerBusReady**	**tempIndex**	**MAX_VELOCITY**
Prefix	--	--	-- (default) "inst" (param-instance)	"stat" (default) "inst" (multi-instance) -- (in global data block)	"temp"	--
Casing	camelCasing	camelCasing	camelCasing	camelCasing	camelCasing	UPPER_CASING

Tag table	*PLC tag*	*User constant*
	lightBarrierLeft	**MAX_BELTS**
Prefix	--	--
Casing	camelCasing	UPPER_CASING

Programming styleguide for SIMATIC S7-1200/ S7-1500 in TIA Portal

- Unique, meaningful identifiers in English
- Only the characters a-z, A-Z, 0-9 and _
- Maximum 24 characters per identifier
- Array: axesData [0..#MAX] of type…
- Library: Name max. 8 chars; prefix "LExample_"

Object		Prefix	Casing
Project	**AssemblyLine**	--	PascalCasing
Library	**LCom**	"L"	PascalCasing
Organization block	**Main**	--	PascalCasing
Function block	**HeatTank**	--	PascalCasing
Function	**CalculateTime**	--	PascalCasing
Global data block	**MachineData**	--	PascalCasing
Single instance data block	**InstHeater**	"Inst"	PascalCasing
Technological object	**HeatingAxis**	--	PascalCasing
PLC tag table	**Sensors**	--	PascalCasing
Watch/Force table	**MachineState**	--	PascalCasing
Trace	**ConveyorSpeed**	--	PascalCasing
Measurement	**Temperature**	--	PascalCasing
PLC alarm text list	**ConveyorAlarms**	--	PascalCasing
Software unit	**Magazine**	--	PascalCasing
PLC datatype	**typeDiagnostics**	"type"	camelCasing
Element in a PLC datatype	**stateNumber**	--	camelCasing

Usual abbreviations (maximum one per identifier)

Min / Max	Minimum / Maximum
Act	Actual, Current
Next / Prev	Next / Previous value
Avg	Average
Sum	Total sum
Diff	Difference
Cnt	Count
Len	Length
Pos	Position
Ris / Fal	Rising / falling edge
Old	Old value
Sim	Simulated
Dir	Direction
Err / Warn	Error / Warning
Cmd	Command
Addr	Address

12 附件

12.1 服务和支持

行业在线支持

你有问题还是需要支持？

通过行业在线支持，您可以 24/7 访问完整的服务和支持诀窍以及我们的服务产品。

行业在线支持是提供有关我们产品，解决方案和服务的信息的中心地址。

产品信息，手册，下载，常见问题和应用程序得案例，所有信息仅仅需要点击下几次鼠标即可:
https://support.industry.siemens.com

技术支持

西门子工业的技术支持为您提供各种量身定做的产品，以快速和胜任的方式满足所有的技术要求
-从基本支助开始直至个人支助合同。

向技术支持提出的请求可以通过网页发送:
www.siemens.de/industry/supportrequest

SITRAIN-工业培训

我们在全球范围内为我们的产品和解决方案提供培训，我们以创新的学习方法和定制的概念支持您。

有关提供的培训和课程以及我们的地点和日程安排的更多信息，请访问:
www.siemens.de/sitrain

服务

我们提供的服务包括以下内容:

- 工厂数据服务
- 备件服务
- 修理服务
- 现场和维护服务
- 改装和现代化服务
- 服务计划和合同

有关我们提供服务的详细信息，请参阅我们的服务目录:
https://support.industry.siemens.com/cs/sc

行业在线支持 App

使用“西门子工业在线支持”应用程序，您将获得最佳的支持，在未来该应用程序在苹果 iOS，Android 和 Windows Phone 上均可使用:
https://support.industry.siemens.com/cs/ww/en/sc/2067

12.2 链接和文献

表 12-1

	主题
\1\	西门子工业在线支持 http://support.industry.siemens.com/
\2\	编程风格指南 https://support.industry.siemens.com/cs/ww/en/view/81318674
\3\	标准化指南 https://support.industry.siemens.com/cs/ww/en/view/109756737
\4\	在 TIA Portal 进行库处理的操作指南 https://support.industry.siemens.com/cs/ww/en/view/109747503
\5\	用户自定义文档 https://support.industry.siemens.com/cs/ww/en/view/109755202/114872699275
\6\	SIMATIC S7-1200/S7-1500 的比较列表 https://support.industry.siemens.com/cs/ww/en/view/86630375
\7\	STEP 7 (TIA Portal) 和 S7-1200/S7-1500 的通用函数库 (LGF) https://support.industry.siemens.com/cs/ww/en/view/109479728

12.3 历史版本

表 12-2

版本	日期	更改
v1.0	10/2014	内部评审后的首次发布
v1.1	06/2015	修改和更正
v1.2	10/2016	修改和更正 新增:备忘录
v2.0	05/2020	分类与现代化 • 根据工作流分类 • 扩展性能和设计结构- • • 扩展编程指南 • 修正说明 • 检查备忘单