

AF LIBRARY

Automation Framework V1.2

Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.

Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.

By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.

The Siemens terms of use (<https://support.industry.siemens.com>) shall also apply.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit
<https://www.siemens.com/industrialsecurity>.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under
<https://www.siemens.com/cert>.

目录

1.	新增内容	8
1.1.	新功能	8
1.2.	变更日志	8

2.	简介	9
2.1.	概览	9
2.2.	交付范围	10
2.3.	使用的库	10
2.4.	硬件和软件要求	11
2.5.	用户自定义文档 - 在线帮助	13
2.5.1.	用户自定义文档的目录	13
2.5.2.	Code2Docu	14
2.6.	编程规范	15
3.	AF 指南	17
4.	库处理和更新	18
4.1.	库概念	18
4.2.	如何解决库不一致问题	18
4.3.	如何使用提供的库	21
4.4.	如何使用全局库更新项目	24
4.5.	如何更新模板副本	28
5.	硬件配置	30
5.1.	具有 RT/IRT 组态的 PROFINET IO	30
5.2.	PROFINET 拓扑	32
5.3.	时间同步和时间设置	32
6.	PLC 软件架构	37
6.1.	ISA-88 设计	37
6.2.	状态机	40
6.3.	程序结构	44
6.4.	程序执行	48
6.5.	PLC 启动和循环调用序列	50
6.6.	命令和数据流	52
6.7.	软件设计原则	54
6.7.1.	通过块接口传递数据与直接全局访问	54
6.7.2.	PLC – HMI 握手	55
7.	HMI 软件架构	57
7.1.	画面布局	57
7.2.	标题	59
7.3.	导航	63
7.3.1.	主导航	63

7.3.2. 子导航	64
7.3.3. 第三级导航.....	66
7.3.4. 单元导航.....	67
7.3.5. 常规导航命令	70
7.3.6. HMI 项目树导航.....	71
7.4. 机器概述.....	72
7.5. 操作锁定机制	72
8. HMI 中央功能.....	75
8.1. 全局.....	75
8.2. 概览.....	76
8.3. 参数（主导航）	76
8.4. 手动.....	78
8.4.1. 手动操作行入门.....	78
8.4.2. 框架演示实现.....	81
8.5. 消息.....	87
8.5.1. 报警.....	87
8.5.2. 审计.....	87
8.6. 诊断.....	91
8.6.1. 系统诊断.....	91
8.6.2. PROFINET 诊断	92
8.6.3. 接口 Booleans	93
8.6.4. 驱动器诊断.....	96
8.7. 状态机模型.....	109
8.8. 网络服务器.....	109
8.9. 书签.....	110
8.10. 设置.....	112
9. 单元.....	116
9.1. 常规结构.....	116
9.2. 接口和块.....	119
9.3. HMI 画面.....	122
9.4. 模式和状态管理器	123
9.5. 堆叠灯	125
9.6. 演示单元.....	126
9.7. 如何添加新单元？	127
9.8. 如何移除 HMI 上的单元	129
9.9. 如何调整 HMI 的大小.....	135

10.	设备模块.....	139
10.1.	常规结构.....	139
10.2.	接口和块.....	141
10.3.	Hmi 画面.....	143
10.4.	与单元链接.....	145
10.5.	直接控制.....	145
10.6.	EM – EM 关联.....	146
10.7.	作业控制.....	146
10.8.	顺控.....	147
10.9.	EM 模板/示例	150
10.10.	EM General	152
10.11.	EM 演示实例 – 料斗	153
10.12.	EM 演示实例 – 传送带	154
10.13.	EM 演示实例 – 高性能运动.....	156
10.14.	如何添加新设备模块.....	157
11.	控制模块.....	160
11.1.	接口.....	160
11.2.	组态.....	160
11.3.	操作模式.....	161
12.	安全.....	162
12.1.	基本信息.....	162
12.2.	安全和 ISA-88.....	165
12.3.	标准程序和安全程序之间的数据交换	165
12.4.	安全程序.....	167
12.5.	演示.....	169
13.	运动控制.....	170
13.1.	概述.....	170
13.2.	Motion 软件单元.....	171
13.3.	工艺对象与 Sina 功能块	172
14.	通信.....	174
14.1.	机器-MES 通信.....	174
14.2.	机器-机器通信	177
14.3.	与第三方系统通信	180
15.	过程诊断.....	181
15.1.	概述.....	181

15.2.	先决条件.....	182
15.2.1.	PLC 设备设置.....	182
15.2.2.	多语言支持.....	182
15.2.3.	ProDiag 函数块.....	184
15.2.4.	ProDiag 许可证 – PLC.....	184
15.2.5.	启用确认.....	184
15.2.6.	PLC 监控和报警.....	185
15.2.7.	常见报警类别设置.....	185
15.2.8.	系统诊断设置.....	186
15.2.9.	监控设置.....	186
15.2.10.	HMI 报警类别设置.....	188
15.2.11.	自定义监控组态.....	189
15.3.	诊断原理.....	189
15.3.1.	报警组态.....	190
15.3.2.	报警实现.....	190
15.3.3.	LBC 库块的错误处理	192
16.	仿真.....	195
16.1.	概览.....	195
16.2.	PLC 和 HMI 集成仿真信号	196
16.3.	使用 PLCSIM 进行 PLC 仿真	197
16.4.	使用 SIMIT 仿真系统行为.....	201
17.	用户管理.....	203
17.1.	概述.....	203
17.2.	演示实现.....	205
18.	运行演示.....	208
19.	附录.....	211
19.1.	西门子准则.....	211
19.2.	TIA Portal 选件.....	212
19.3.	用于共享任务的软件一目了然	212
19.4.	驱动器的应用	213
20.	附录.....	214
20.1.	服务和支持.....	214
20.2.	文献链接.....	215
20.3.	变更文档.....	216

1. 新增内容

1.1. 新功能

版本	日期	描述
1.0	2023/12/22	第一版
1.1	2024/06/28	<ul style="list-style-type: none">高级单元控制功能（LUC 库扩展）顺控（Graph、LAD 和 SCL）的增强功能从 CM 到 Unit 的整体报警处理手动操作行多单元支持（>=1 个单元，>= 1 个操作面板）画面选择收藏夹
1.2	2024/9/30	<ul style="list-style-type: none">除 S7-PLCSIM Advanced 外，还支持使用 S7-PLCSIM V19 进行仿真将功能“LAF_OmacToSequence”更名为“LAF_MapUnitStatusToSequence”，并增加了配置输入，以便更灵活地配置序列行为。

表 1-1 新功能

1.2. 变更日志

查看 [变更文档](#)

2. 简介

2.1. 概览

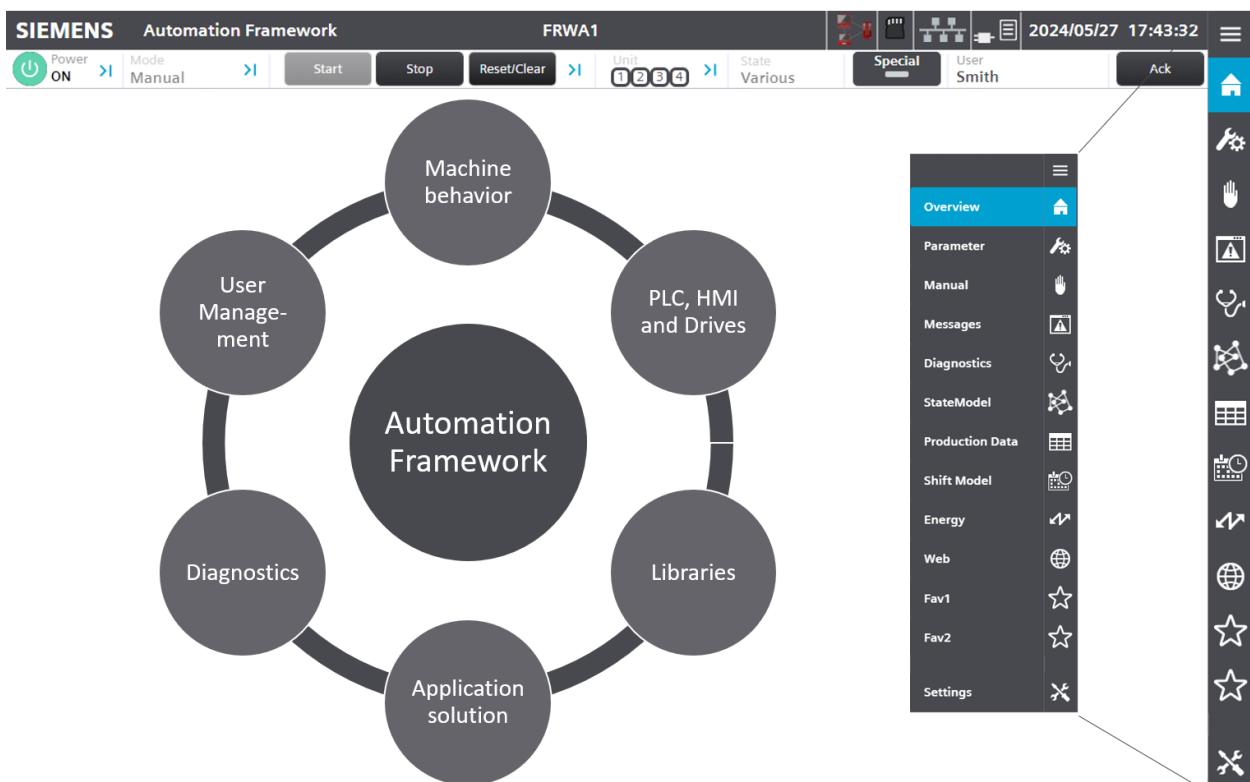


图 2-1 西门子自动化框架

西门子自动化框架（以下称为 AF）的核心产品是预组态 TIA Portal 项目中的 TIA Portal 库、应用示例、硬件配置和编程最佳实践的综合集合，它们构成了一个有机整体。它提供了开发通用自动化应用程序的基本功能。它配备了一个完全组态的操作面板，其中包含主要的导航和操作功能。在此基础上，用户可以使用库中的其他 PLC 和 HMI 对象通过模块化方式轻松构建和扩展自己的项目。

该项目介绍了一种（并非规定版本）直观的 TIA Portal 项目结构规划的最佳实践，并提供了可立即投入使用的 PLC、运动控制和 HMI 编程的应用示例。

本文档介绍了 TIA Portal 项目 Start_Up_AF_Library（简称 AF 项目）及其库的内容和结构。用户可以随意修改该项目或从空项目重新开始，以避免在应用中出现未使用的代码。

主要特性

- 模块化 PLC 和 HMI TIA Portal 项目符合 ISA-88 的物理模型
- 轻松实现 OMAC PackML 兼容的模式和状态管理器
- 预组态的操作面板和扩展组态的位置和方式的指南
- 轻松集成和处理库和更新
- 各种 TIA Portal 库以及应用解决方案（示例）
- 通过系统功能 ProDiag 实现统一的过程诊断
- 使用 HMI 模板进行手动操作，例如执行点动操作

目标

- 适合各种机器类型的整体 TIA Portal 项目和库
- 分享如何实施各种应用用例的指导和最佳实践
- 通过使用其他西门子技术产品、开箱即用的库、应用示例、模板以及集成它们的方法，轻松进行扩展

机器制造商和设备最终用户的收益

- 模块化提高了产品重用性和投资回报率
- 设备隔离增加灵活性
- 不同机器类型的快速实施和协调操作
- 灵活管理更新、功能扩展和变更请求
- 预组态的运营关键绩效指标 (KPI)
- 统一的过程和系统诊断可加快调试速度并最大限度减少生产停机时间

注

该 AF 项目的目的是用作各种机器类型的 TIA Portal 项目模板。它从未宣称是一个涵盖各个方面的开箱即用解决方案。用户仍然需要开发自己的应用程序，实施所需的用例，并在必要时整合更新和补丁。
对于现有的机器代码实现，用户可以在适当的情况下导入和利用 AF 库的元素。

必备知识

为更方便理解手册内容，需要具备以下方面基础知识：

- 自动化工程

- SIMATIC 和 SINAMICS 硬件
- SIMATIC STEP 7 (TIA Portal)
- SIMATIC WinCC Unified (TIA Portal)
- SIMATIC Safety (TIA Portal)
- SINAMICS Startdrive (TIA Portal)

- 机器和设备诊断

- 工业通信与网络
- OPC UA

注

请注意，使用 TIA Portal 项目需要最新的 Unified Panel 映像版本（从 V19.0.0.2 起）。

2.2. 交付范围

AF 1.2 版本包括：

- 本用户文档 “109817223_Automation_Framework_DOC_V1_2.pdf”
- TIA Portal 项目 “109817223_Automation_Framework_PROJ_V1_2.zap”
- TIA Portal 项目 “109817223_Automation_Framework_Startup_PROJ_V1_2.zap”

2.3. 使用的库

AF 项目中使用以下 TIA Portal 库：

AF 全局库

AF 全局库包含推荐用于 AF 的所有子库、模板副本和用户文档。即将发布的版本、补丁和更新将通过此全局库提供。基于 TIA Portal 标准功能的更新过程在[库处理和更新](#)一章中进行了介绍。

库名称	描述	链接
LGF	通用函数库 (LGF) 扩展了 TIA Portal 中用于 PLC 编程的 STEP 7 指令（数学函数、时间、计数器等）。该库可以不受限制地使用，并包含 FIFO、搜索功能、矩阵计算、astro 计时器等功能。	SIOS-ID: 109479728
LPML	OMAC PackML 库 (LPML) 为组态和使用 OMAC 兼容的 SIMATIC 模式和状态管理器奠定了基础。	SIOS-ID: 109821198
LUC	单元控制库 (LUC) 提供的函数块可简化 LPML OMAC 状态机处理、预处理操作员或远程命令（例如通过 OPC UA），并提供堆叠灯实现。它使用 LPML 中的块和数据类型，并扩展其功能。	SIOS-ID: 109974940
LAF	自动化框架库 (LAF) 提供了可轻松实现 ISA-88 标准和常用功能的块。它使用 LUC 库中的块和数据类型，并扩展其功能。	SIOS-ID: 未规划
LDrvSafe	安全驱动控制库 (LDrvSafe) 包括用于实现安全应用的块，例如，通过 PROFIsafe 轻松控制 SINAMICS 安全功能以及故障安全直径检测，最高可达安全完整性等级 2 (EN 62061) 和性能等级 d 类别 3 (EN ISO 13849-1)。	SIOS-ID: 109485794
LSafe	经过 TÜV 认证的 LSafe 库可用于实现机电或电子传感器和执行器的基本安全功能。该库有助于验收您的应用软件，因为它可以基于通过测试的模块。	SIOS-ID: 109793462
LSNTP (LCom)	使用 SIMATIC S7 CPU 作为 SNTP 服务器，可以灵活、简单地同步系统和子系统，例如，获取系统级错误消息和记录数据的时间戳。	SIOS-ID: 109780503
LAxisCtrl	该库提供了一个轴函数块，该函数块具有非常丰富的功能，可用于简化轴的控制。它在 LBC 库块中进行轴控制或独立使用。	SIOS-ID: 109749348
LBC	“基本控制库” (LBC) 提供基本控制功能，这些控制功能根据西门子编程风格指南和“PLC Open”指南进行标准化编程。	SIOS-ID: 109792175
LSicar V5	“LSicar” 库包含用于生产数据、班次模型和诊断的有用且经过充分验证的功能。	SIOS-ID: 109804254
LPD	“PLC 数据类型库” (LPD) 包含 PLC 数据类型，用于描述地址空间的数据结构以及外围设备/技术模块和 PROFIdrive 驱动器的数据记录。	SIOS-ID: 109482396

表 2-1 AF 项目中使用的库

2.4. 硬件和软件要求

AF 项目基于以下组件开发：

硬件

- SIMATIC S7-1500 (F/ TF) 固件版本 V3.0.3 或更高版本 – 有关更多信息，请参见 [SIOS-ID: 109478459](#)
- WinCC Unified Panel 显示屏尺寸 12 英寸（面板映像 V19.0.0.2 以及更高版本，[SIOS-ID: 109825605](#)）
- SINAMICS S200、S210 和 G120
- ET 200SP

软件

- TIA Portal V19 Update 3 或更高版本
- 用于 PLC 工程的 SIMATIC STEP 7 Professional V19 Update 3 或更高版本
- 用于 HMI 工程的 SIMATIC WinCC Unified V19 Update 3 或更高版本
- 用于驱动器组态和调试、PLC 连接、诊断、优化和参数化的 SINAMICS Startdrive Basic V19 SP1 或更高版本
- TIA Portal 插件 Code2Doc^[4], 用于生成 PLC-Block 文档

可选产品

- S7-PLCSIM V19 或更高版本, 用于仿真实体 PLC 固件
- S7-PLCSIM Advanced V6.0 或更高版本, 用于模拟带虚拟网络适配器的实体 PLC 固件
- SIMATIC WinCC Unified PC Runtime V19 Update 2 或更高版本, 用于仿真 HMI 应用程序
- SIMATIC Visualization Architect (SiVarc) 用于自动生成“即用型”画面
- TIA Portal Test Suite 用于自动化代码单元测试
- SINAMICS Startdrive Advanced 用于安全验收和安全启动测试、测量功能和长期轨迹
- SIMIT 用于模拟驱动器、传感器、执行器、I/O 等的行为以及与外围设备和 PLC 的连接, 以验证 PLC 程序的功能和通信。
- NX Mechatronics Concept Designer (MCD) 2406 或更高版本, 用于基于三维的物理仿真, 以规划和验证机器序列、碰撞检测和虚拟调试

许可证

AF 项目使用以下许可证:

必选	有关订单和版本信息, 请参见当前的 AF 组件发布列表	必须交付给最终用户的许可证
是	SIMATIC STEP 7 Professional V19	
是	SIMATIC S7, 安全编程工具	
是	STEP 7 Safety Advanced	
否	SIMATIC S7-PLCSIM V19	
否	SIMATIC S7-PLCSIM Advanced V6.0	1x SIMATIC S7-PLCSIM Advanced 许可证
是	SIMATIC WinCC Unified V19 Comfort Engineering	
否	SIMATIC WinCC Unified PC Runtime V19	1x SIMATIC WinCC Unified PC (10k) RT 许可证
是	SINAMICS Startdrive Basic V19	
否	SINAMICS Startdrive Advanced V19	1x SINAMICS Startdrive Advanced 工程许可证
是	ProDiag for PLC Runtime	每个 CPU 需要 1 x ProDiag unlimited Runtime 许可证
否	ProDiag for HMI (PLC 代码查看器)	每个控制屏幕需要 1 x SIMATIC ProDiag for WinCC Unified Runtime Controls 许可证
否	OPC UA	每个 CPU 需要 1 x OPC UA Runtime 许可证, 许可证类型 (小型、中型、大型) 取决于接口类型
否	SIMIT	1x SIMIT 工程许可证, 许可证类型 (S、M、L、XL) 取决于模拟范 围
否	NX MCD	1x MCD Designer 许可证(模型编辑和虚拟调试) 或 1x MCD Player 许可证(仅虚拟调试)

表 2-2 AF 项目许可证

注 OPC UA 和 HMI power tag 许可证数量可能因特定应用而异。根据实际需要，确定相应的许可证数量。
在 AF 项目中使用 PLC Runtime 中的 ProDiag 可简化用户的工程设计。此外，它也可以很容易地被应用解决方案所取代。

2.5. 用户自定义文档 - 在线帮助

在使用本产品的过程中，可以在项目或库中创建自己的内容，例如，块、变量或库类型。虽然提供的帮助系统 (F1) 中描述了 TIA Portal 的工作原理，但对于自己创建的内容，没有提供任何帮助。

TIA Portal 提供了一个系统功能，用于打开与块相关的文档 (Shift+F1)。可以创建用户自定义文档，向其他员工解释项目和功能如何工作，或者如何使用各个库类型或块。

注 用户自定义文档不会与类型一起复制到另一个库。需要手动将用户自定义类型文档复制到相应的目录中。
有关使用用户自定义文档的其他帮助，请参见 TIA Portal 文档中的“使用用户自定义文档”部分。

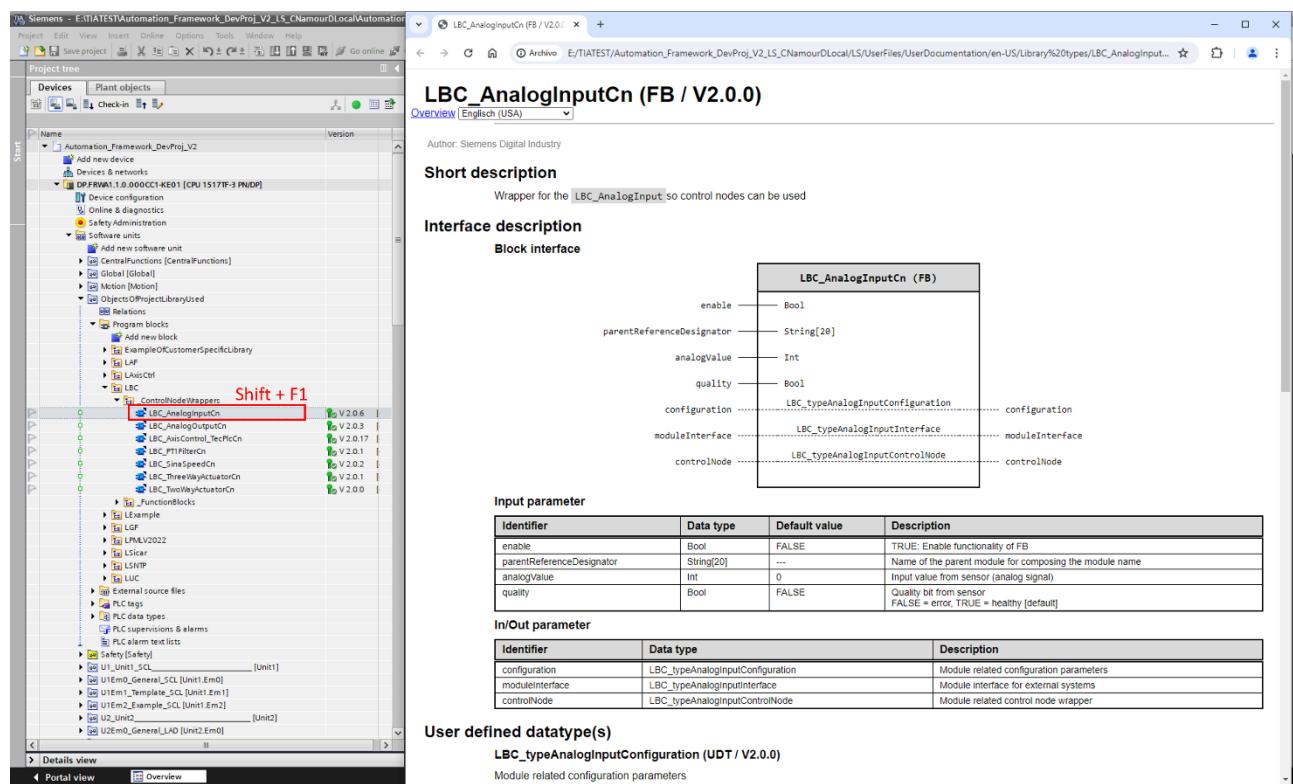


图 2-2 如何打开库程序块的在线帮助

2.5.1. 用户自定义文档的目录

用户自定义文档可以保存在以下目录中：

TIA Portal 项目文件夹

将文档保存在 [项目文件夹]\UserFiles\UserDocumentation\[语言] 下。通过这种方式，文档可以与 TIA Portal 项目一起转发。

全局库的目录

将文档保存在 [全局库]\UserFiles\UserDocumentation\[语言] 的目录中。这样，用户自定义文档就可以与全局库一起转发。

硬盘驱动器或网络驱动器上的中心目录

可以将文档存储在硬盘或网络驱动器上的中心目录中。通过这种方式，可以访问来自不同项目的文档或通过网络驱动器共享文档。用户自定义文档的目录路径必须在 TIA Portal XML 组态文件中指定，或者通过 TIA Portal 设置指定。

2.5.2. Code2Docu

在 AF 中，用户文档在每个块的代码部分编写（LAD 块为程序段 1 和 2 的多语言备注，SCL 块为区域文件头信息和描述中的备注）。

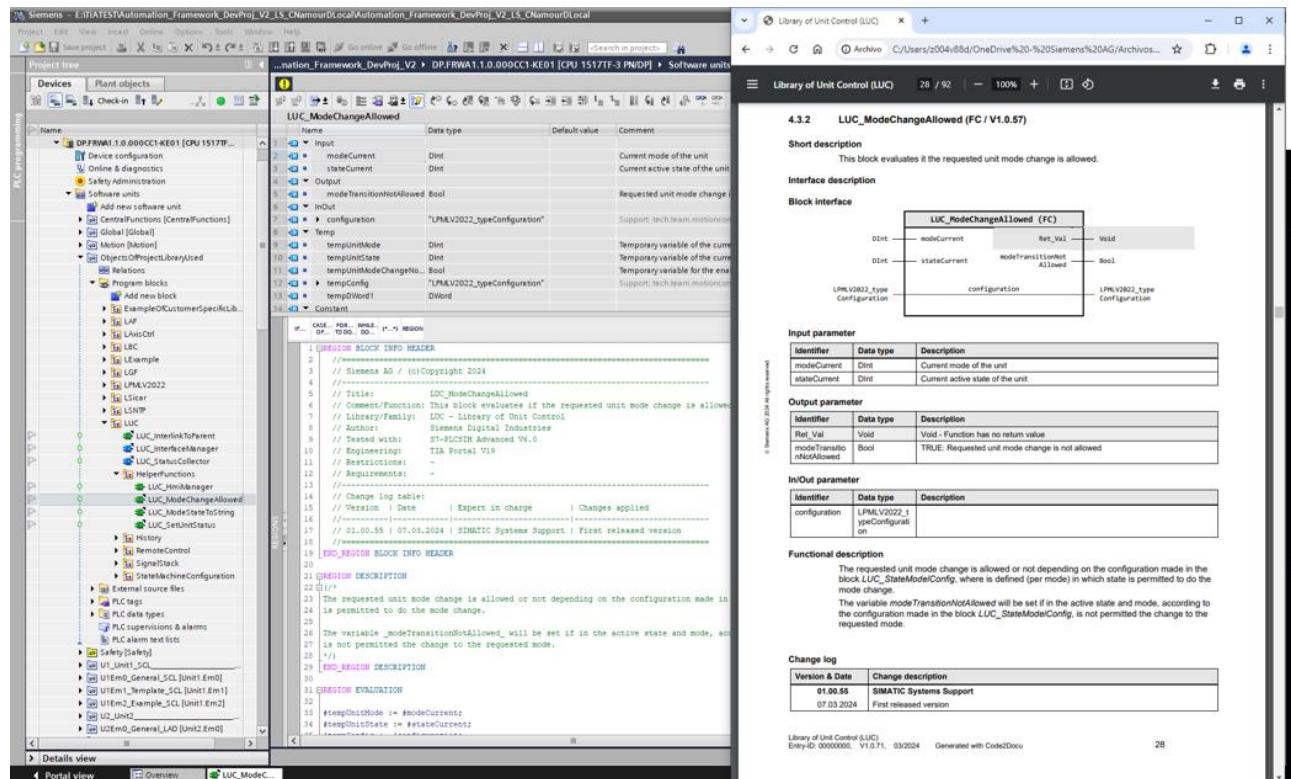


图 2-3 使用插件 Code2Docu 生成的块文档示例

借助 TIA Portal AddIn Code2Docu^[4]，可以自动为每个块创建所有使用语言的文档。

Code2Docu Add-In - Process flow

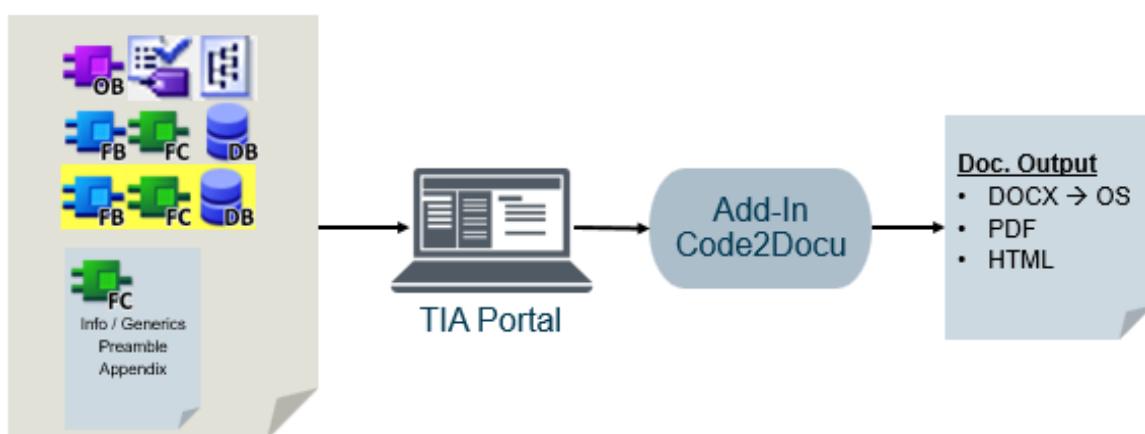


图 2-4 Code2Docu – 流程

2.6. 编程规范

AF 遵循西门子 S7-1200/S7-1500 编程指南的约定以及西门子编程规范指南，西门子安全编程指南和 WinCC Unified 工程组态指南规范。



有关编程风格指南的更多信息，请访问以下链接：
[西门子准则](#)

编程风格指南

除了这些准则外，还需要遵守以下规则：

- **GL003 规则：提供所有项目语言的文本。**

所有项目文本必须以英文和项目中使用的所有其他语言提供。虽然 AF 项目支持中文、英语（英国）、法语、德语、意大利语、韩语和西班牙语，但这些文本目前仅提供英语（美国）版本。用户负责编辑和提供其特定项目语言所需的翻译。

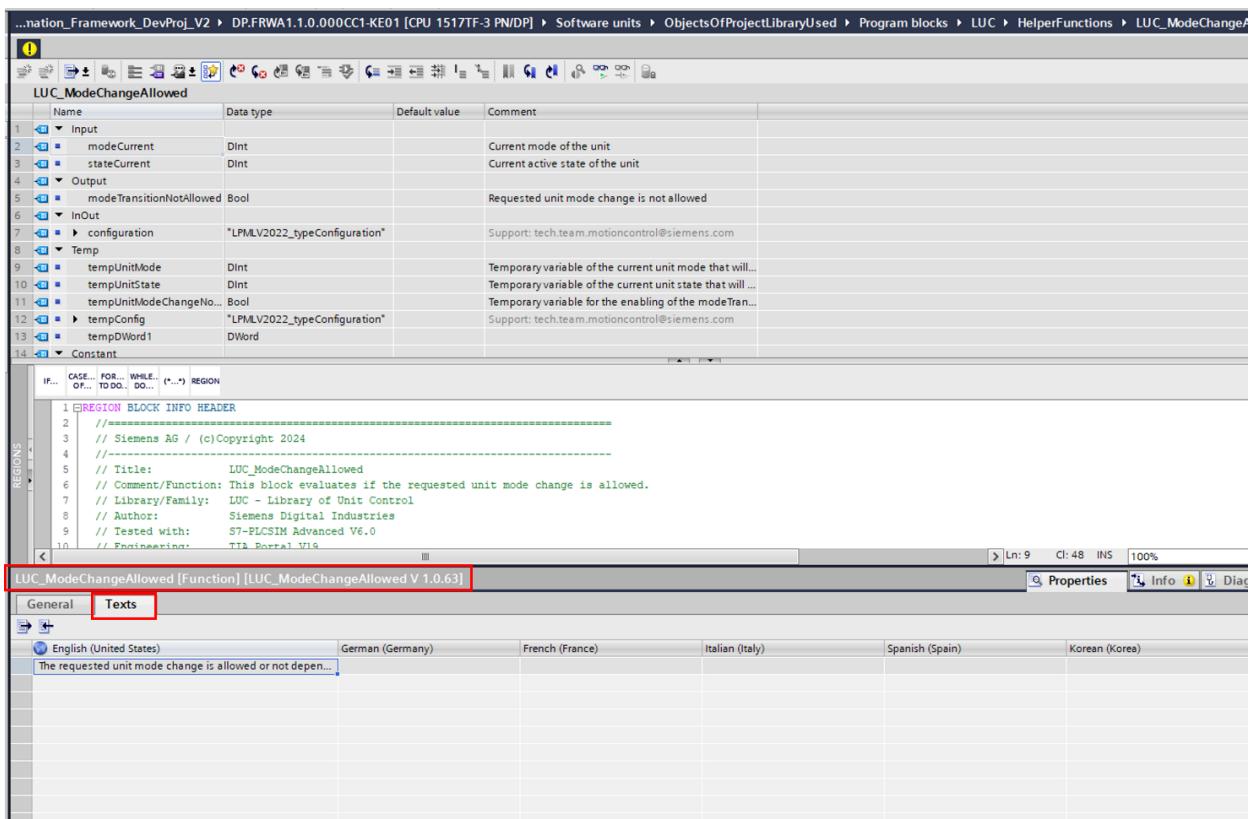


图 2-5 在块编辑器中，文本及其翻译可以在“文本”选项卡中轻松管理

- **NF002 规则：使用便于理解的注释和属性**

注释和属性字段应使用项目的当前语言填写便于理解的信息。尽管 AF 项目支持中文、英语（英国）、法语、德语、意大利语、韩语和西班牙语，但这些文本目前仅以英语（美国）提供。用户需要添加必要的项目语言的翻译。

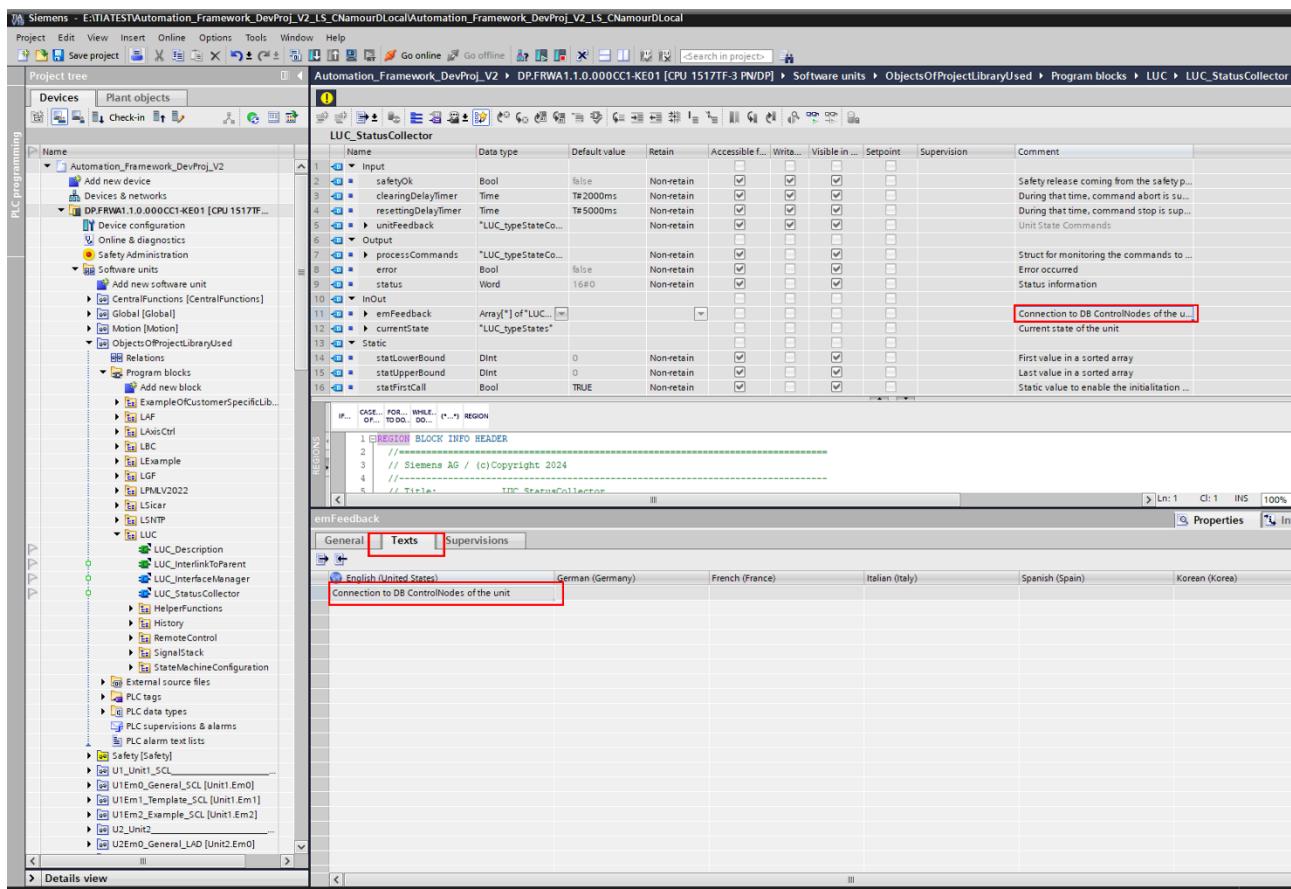


图 2-6 在块编辑器中轻松管理和编辑文本和翻译

• NFO05 规则：使用帕斯卡命名法

TIA Portal 对象（包括块、软件单元、工艺对象、库、PLC 变量表、PLC 报警文本列表、监控表和强制表、轨迹和测量）的标识符均使用帕斯卡命名法设置格式。在 AF 项目中，与帕斯卡命名法的不同之处在于，通过使用连续的大写字母来标识一个对象。

需要打破该规则，以确保对象的标识符清晰易懂。打破该规则的原因是为了在使用广泛认可的缩写（如 LAD、KOP 和 FUP）时保持清晰度，并引入新的缩写（例如用于代表 LibraryAutomationFramework 的 LAF），以确保它们易于识别。例如，根据规则，块“LAF_SequenceManagerLAD”应写为“Laf_SequenceManagerLad”，但这会产生一个令人困惑的标识符。严格遵循规则将导致标识符的可读性降低。

编程指南

强烈建议遵守以下编程规则：

- 不要使用位存储器/标记，而是定义全局数据块 (DB)。
- 禁用块的“评估 ENO”，以提高 LAD/FBD 中的性能。
- 直接在对象 (FC、FB、UDT) 内部添加监控，以发挥类型实例概念的优势。
- 仅通过块接口 (In、Out、InOut) 交换变量，以确保块的可重用性。
- 优先使用 CASE 指令，其次是 IF.. ELSEIF.. ELSE 或循环。
- 使用用户自定义的数据类型 (UDT) 而不是 STRUCT 数据类型。

注

TIA Portal 在一个数据块中最多支持 252 个结构 (STRUCT 数据类型)。

3. AF 指南

本文档介绍如何使用自动化框架。自动化框架使用的其他指导原则请参见[西门子准则](#)章节。

4. 库处理和更新

本章介绍如何以最有效的方式使用 TIA Portal 库。介绍了一些初始操作，以便在项目阶段为库更新做准备，而不会影响自定义代码。AF 库和 AF 项目中的所有类型都以“写保护”方式提供，以防止对原始元素进行更改并确保可更新性。

4.1. 库概念

使用 TIA Portal 库时采用两个概念：类型和模板副本。

注 有关库的更多信息，请参见 [TIA Portal 中的库处理指南](#)。

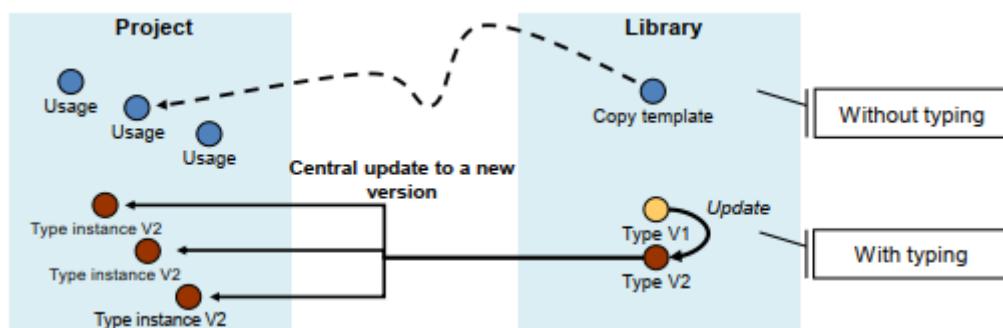


图 4-1 使用库副本与库类型的差异

使用类型

在库中将对象存储为类型时，此对象的任何实例都保持与其原始库类型的关系。类型支持版本控制，并在库对象中实现集中更改。对库类型的更改可以在整个项目及其所有实例中进行更新。在此过程中，只需进行单点更改。

使用模板副本

将对象存储为模板副本时，任何实例都只是此 TIA Portal 库对象的副本。副本既不保持与其源的关系，也不支持版本控制或集中编辑。

因此，建议在条件允许的情况下使用库类型。

4.2. 如何解决库不一致问题

使用嵌套库类型时，对源类型的更改可能会影响其依赖类型。TIA Portal 库的状态将通知用户存在的不一致问题。在从全局库运行项目更新之前，解决这些不一致问题非常重要。

注 建议从“右键单击类型”>“修复不一致”开始，然后根据可用选项“使用默认版本”或“高级编辑类型”来确定默认版本的优先级。更详细的说明，请参见 TIA Portal 的库处理指南。

<https://support.industry.siemens.com/cs/de/en/view/109747503>

使用库管理

有时，需要以下用户交互来解决库不一致问题。右键单击项目库中的“类型”文件夹，然后从快捷菜单中选择“库管理”。不一致问题以黄色突出显示。

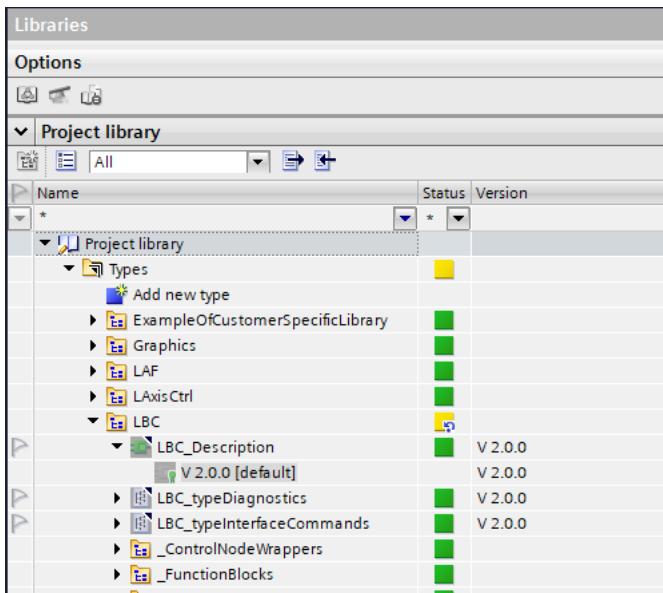


图 4-2 库管理

以下说明介绍了一个库不一致示例以及如何解决它们：

同时出现两个不一致的用例：死锁自动解决

- 最新的 UDT 版本 (V0.0.5) 不是默认版本。
- 当前默认版本 (V0.0.4) 未使用“唯一的”默认类型。

Project library > Types > LBC > LBC_AnalogInputCn					
Type	Status	Version	Uses	Path	
LBC_AnalogInputCn	V 0.0.4	V 0.0.5		LBC_AnalogInputCn	Project library\Types\LBC\FunctionBlocks\LBC_AnalogInputV 1.1.12 [default]
V 0.0.5				LBC_typeAnalo...	Project library\Types\LBC\AnalogSignals\input\LBC_typeAnalogInputConfigurationV 1.1.0 [default]
				LBC_typeAnalo...	Project library\Types\LBC\AnalogSignals\input\LBC_typeAnalogInputInterfaceV 1.1.3 [default]
				LBC_typeAnalo...	Project library\Types\LBC\AnalogSignals\input\LBC_typeAnalogInputControlNodeV 0.0.2 [default]
V 0.0.4 [default]	V 0.0.4			LBC_typeAnalo...	Project library\Types\LBC\AnalogSignals\input\LBC_typeAnalogInputConfigurationV 1.1.0 [default]
				LBC_typeAnalo...	Project library\Types\LBC\AnalogSignals\input\LBC_typeAnalogInputInterfaceV 1.1.3 [default]
				LBC_typeAnalo...	Project library\Types\LBC\AnalogSignals\input\LBC_typeAnalogInputControlNodeV 0.0.2 [default]
				LBC_AnalogInp...	Project library\Types\LBC\FunctionBlocks\LBC_AnalogInputV 1.1.13

图 4-3 库管理

显示的信息有助于理解当前的不一致问题：

- 使用哪些类型及其文件夹作为可单击的超链接以打开目标。
- 使用的每种类型的版本。
- 哪个版本处于状态 [默认]。

分辨率：将 HMI 类型的接口更新为默认（最新）版本。

- 打开 HMI 类型 - 右键单击类型并选择“编辑类型”。
- 转到变量接口，检查使用的是哪个 UDT 版本。

Project library > Types > LBC > WinCC_Unified > AnalogSignals > LBC_AnalogMonitoring > V 2.0.2

Name	Data type	User data type structure
moduleInterface	PLCUDT	LBC_typeAnalogInputInterface V 2.0.2
<Add new>		

图 4-4 变量接口

- 检入“库管理”，这是默认版本。

Type	Version	Instances in the project	Path
LBC_typeAnalogInputInterface	V 2.0.2		
V 2.0.2 [default]	V 2.0.2	LBC_typeAnalogInputInterface	Automation_Framework
V 2.0.0	V 2.0.0	No other types	

图 4-5 库管理

- 在 HMI 类型的变量接口中，将默认版本连接到 UDT。

Project library > Types > LBC > WinCC_Unified > AnalogSignals > LBC_AnalogMonitoring > V 2.0.3

Name	Data type	User data type structure
moduleInterface	PLCUDT	LBC_typeAnalogInputInterface V 2.0.2
<Add new>		

图 4-6 更新接口 UDT

- 现在，可以将 HMI 类型发布为新的默认版本。
- 请始终设置两个复选框以保持一致。

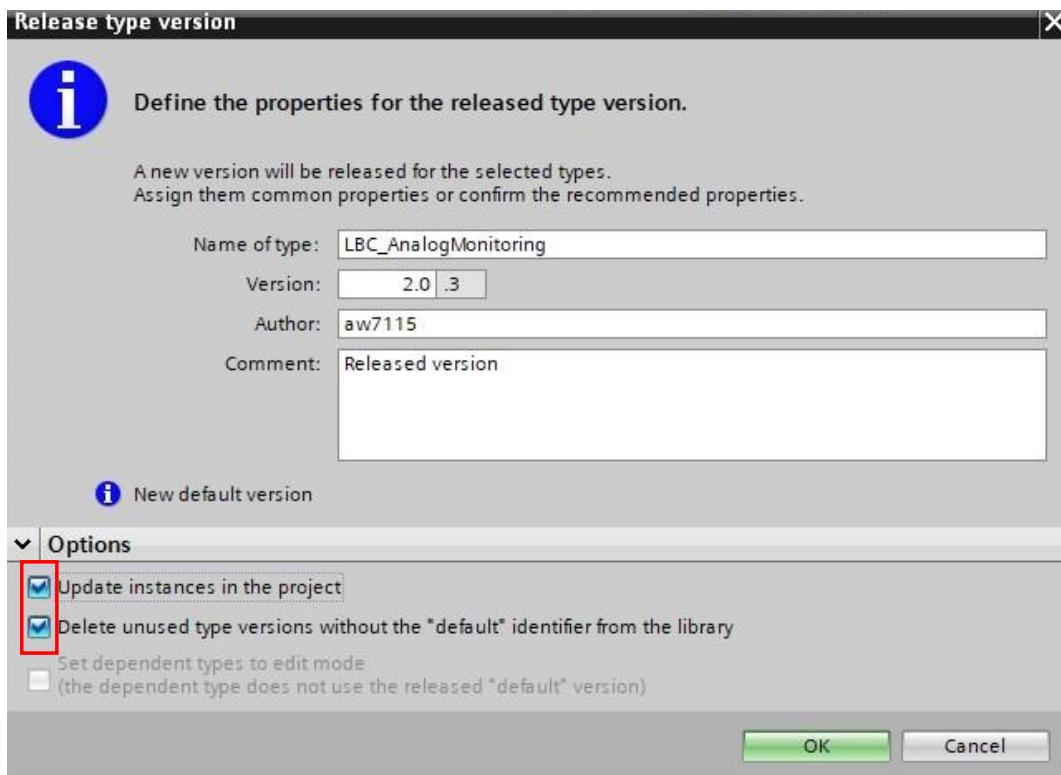


图 4-7 发布类型

注 如果发生其他库不一致，则需要以类似的方式进行修复。

4.3. 如何使用提供的库

AF 带有一个全局库，其中提供的大多数对象（而不是全部）都是类型化对象。强烈建议用户根据自己的需要重用和调整所提供的功能，但为了确保可更新性，理解以下信息非常重要。

交付的类型

按交付时的状态，使用类型对象，不要修改它们。

- 用户修改将被此类型的下一个库更新（较新版本）覆盖。
- 由于版本冲突，下一次库更新可能不会自动运行。

修改交付的类型/创建自己的类型（在 AF 库内容中）

所有 AF 库类型都受到保护，这就像写保护一样。

这些块不能复制或修改。如果需要基于此块创建自己的派生类型，可以在 SIMATIC 在线支持页面中下载特定库的未受保护版本 - 请参见“[使用的库](#)”一章以查找链接。下载后，请按照“[如何使用全局库更新项目](#)”中的后续步骤进行操作。

用例 - 修改交付的类型/创建自己的类型（常规）

用户收到一个 TIA Portal 库，并希望根据特定需求修改类型（或块），例如调整接口。

推荐的方法是创建希望在库中修改的类型的副本。复制的类型现在是一种新的类型（新的 type-id），可以根据需要进行修改。

按照以下步骤操作：

- 将库中的原始类型复制到 PLC 中。

- 在库中选择类型对象。
- 右键单击并选择“复制类型”。

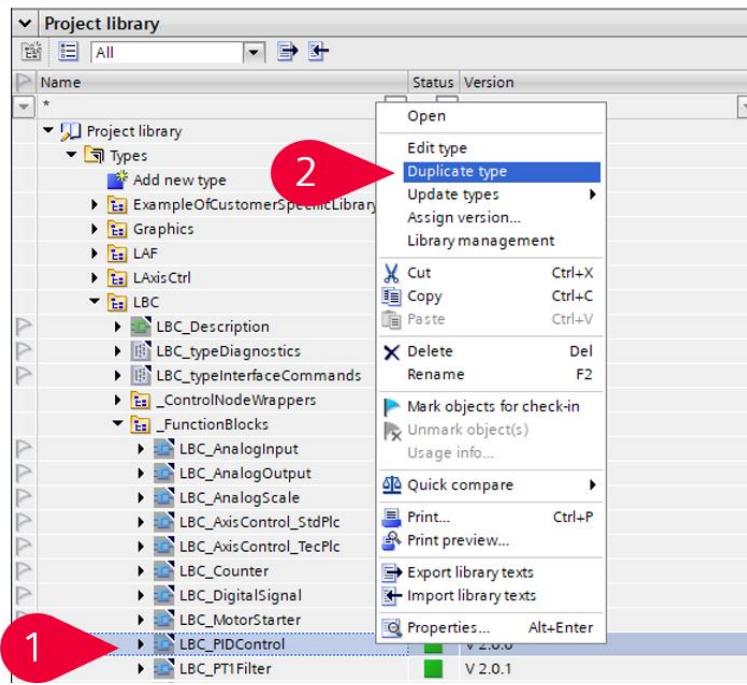


图 4-8 创建类型副本

- 保存类型 - 建议填写新类型的所有属性。

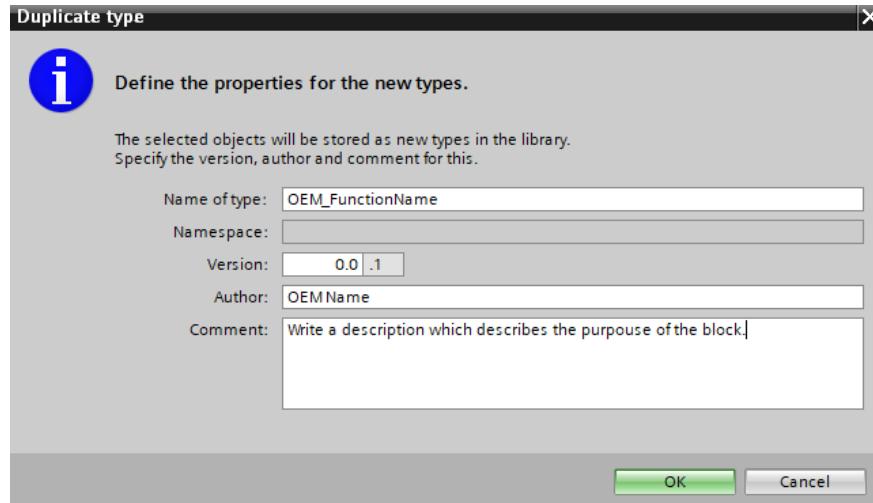


图 4-9 保存类型副本

- 现在，可以根据需要修改新类型，而不会干扰原始类型的更新链。

注

现在，用户需要使自定义类型与原始类型的错误修复或改进保持一致，因此请密切关注所提供库即将进行的更新。

用例 – 在项目中使用自定义类型

如果已经在项目中使用了原始类型，现在要使用新的复制类型，则需要手动将调用（PLC 或 HMI）从原始类型更改为自定义类型。

用例 – 已修改原始类型

如果修改了原始类型，并且收到全局库的新版本作为更新。

- 在开始更新之前，请创建已修改类型的副本，如上所述。
- 现在，可以运行库更新，而不会干扰所做修改。

版本控制

如果发布新的（自定义）类型或版本，则会出现“发布类型版本”框。

请勿更改“默认版本”。新创建的版本是并且应始终是默认版本。

- 始终选中（如果可用）“更新项目中的实例”。
- 始终选中“从库中删除没有“默认”标识符的未使用版本”。从而防止使用错误的版本。
- 始终选中（如果可用）“将依赖项类型设置为编辑模式”。从而防止库出现不一致。

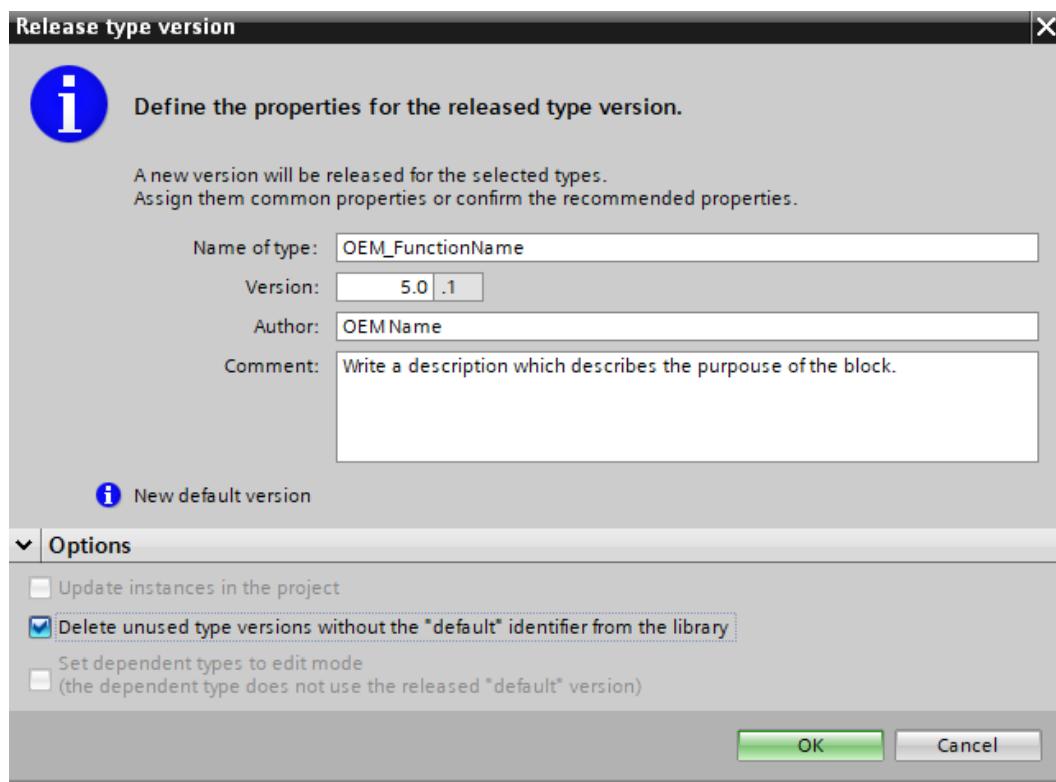


图 4-10 发布所有类型，非默认版本将被删除

- 可以使用“全部发布”命令一次性发布编辑中的所有类型。

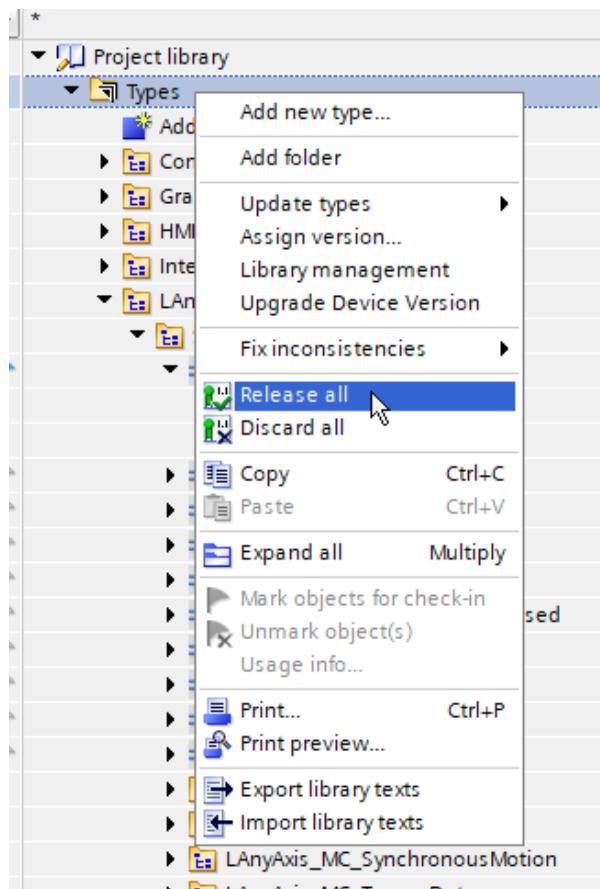


图 4-11 全部发布

如果注意这些提示，可以轻松完成整个库的更新过程，并且不会占用太多时间。

4.4. 如何使用全局库更新项目

全局库提供了一种有效途径，可以将类型化对象的新版本从中央开发项目传输到机器项目。

注 在执行后续步骤之前，请将当前项目另存为备份。

导航到全局库部分

- 转到“库”部分。
- 单击“打开全局库”符号。

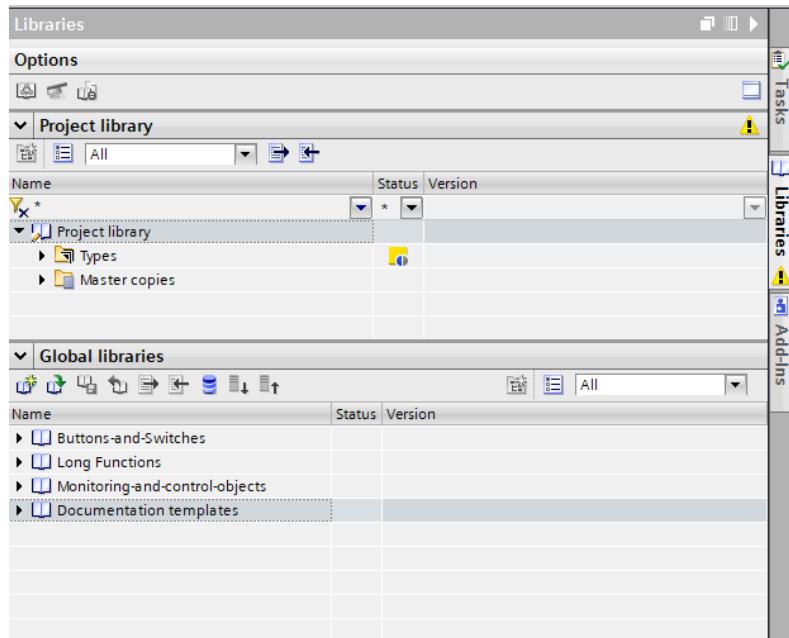


图 4-12 打开全局库

选择并打开全局 AF 库

- 选择“压缩库”作为文件类型。
- 选择要用于更新的库。
- 打开库 - 它存在永久写保护。

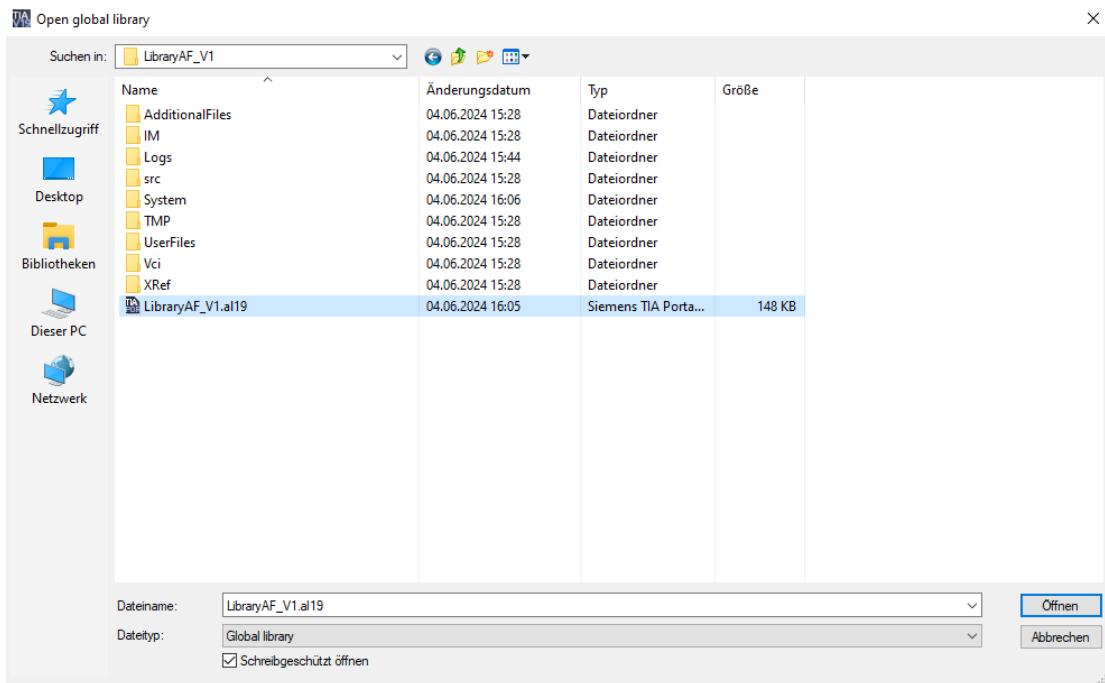


图 4-13 打开全局库

使用新的 AF 库项目更新项目库。

- 右键单击“全局库”。
- 选择“更新类型”选项。
- 选择“库”，以便更新项目的所有类型和实例。

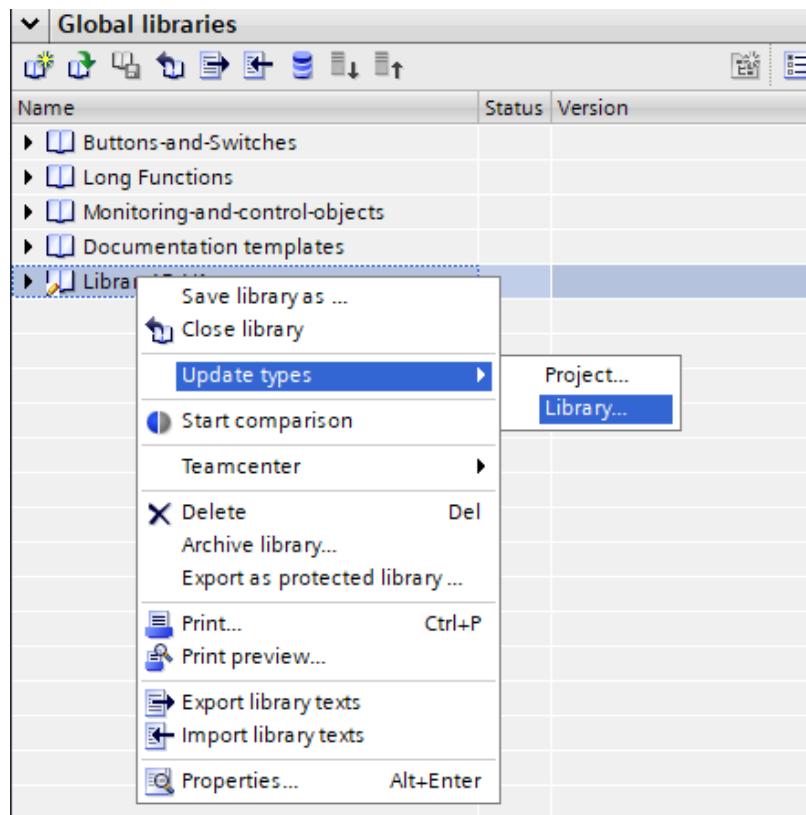


图 4-14 从全局库更新类型

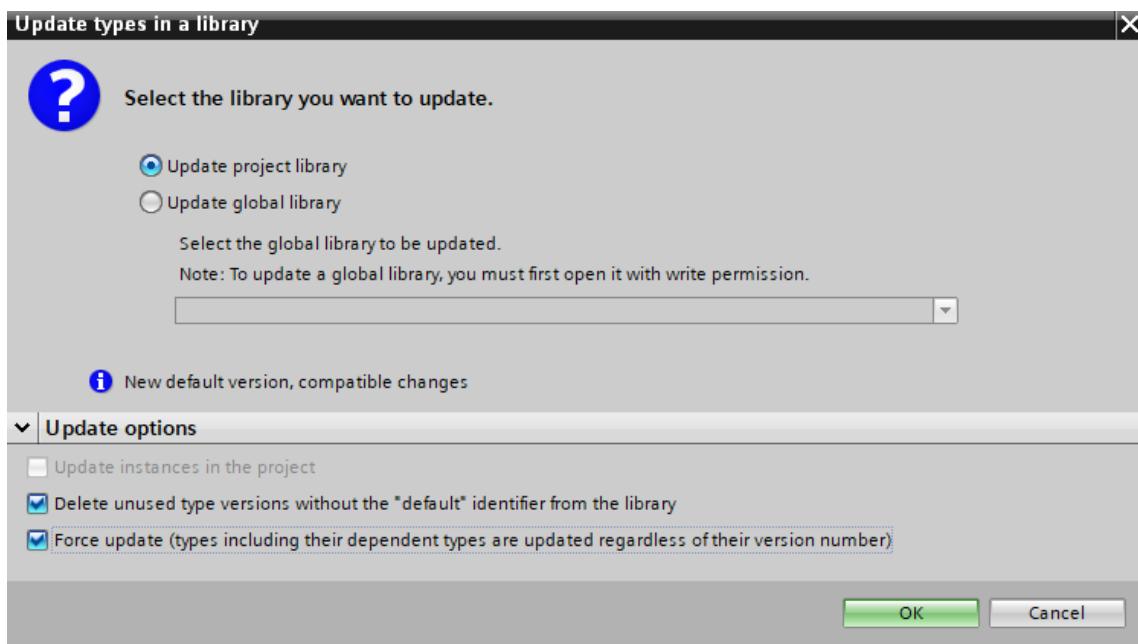


图 4-15 更新项目库中的类型

- 激活“从库中删除没有“默认”标识符的未使用的类型版本”以清理库。
- 激活“强制更新（无论其版本号如何，包括其依赖类型在内的类型都会更新）”。

选择要更新的设备

- 选择项目中应更新的所有设备(1)。
- 激活“从库中删除没有“默认”标识符的未使用类型版本”以清理库(2)。
- 激活“强制更新（类型（包括其依赖类型，无论其版本号如何，都会更新）”(2)。

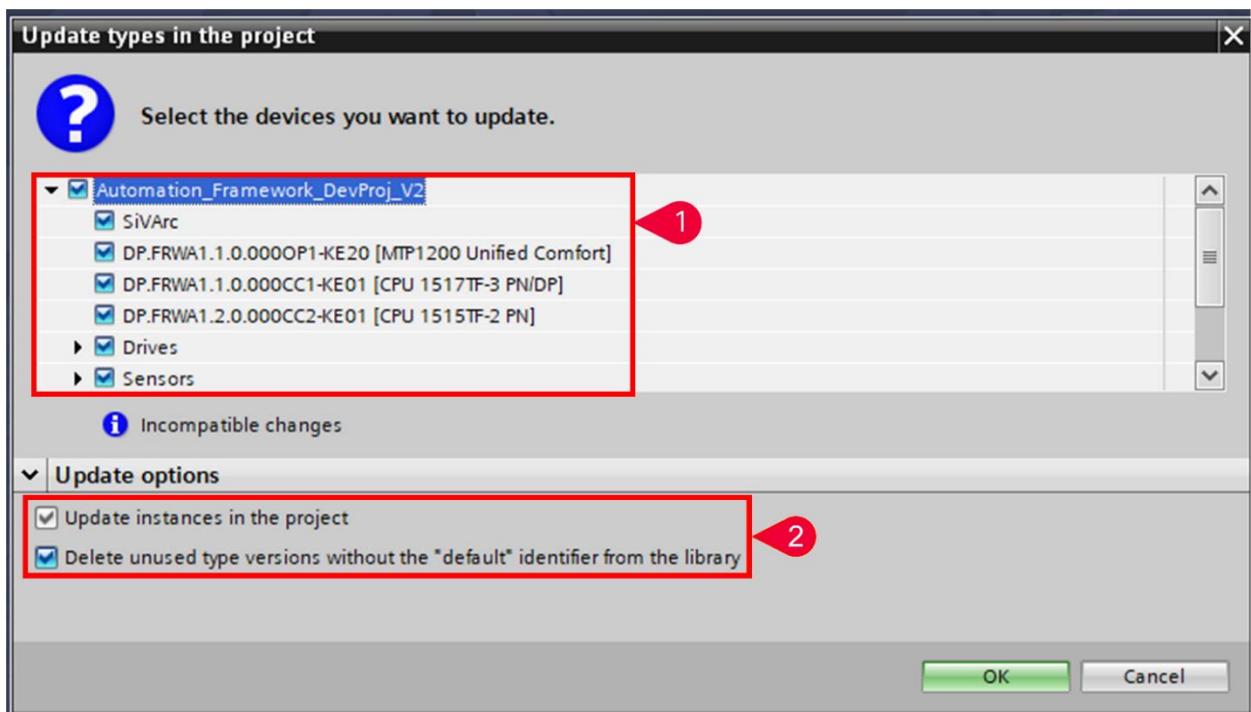


图 4-16 更新项目设备中的类型

4.5. 如何更新模板副本

需要通过复制和粘贴或拖放将模板副本集成到项目库中。

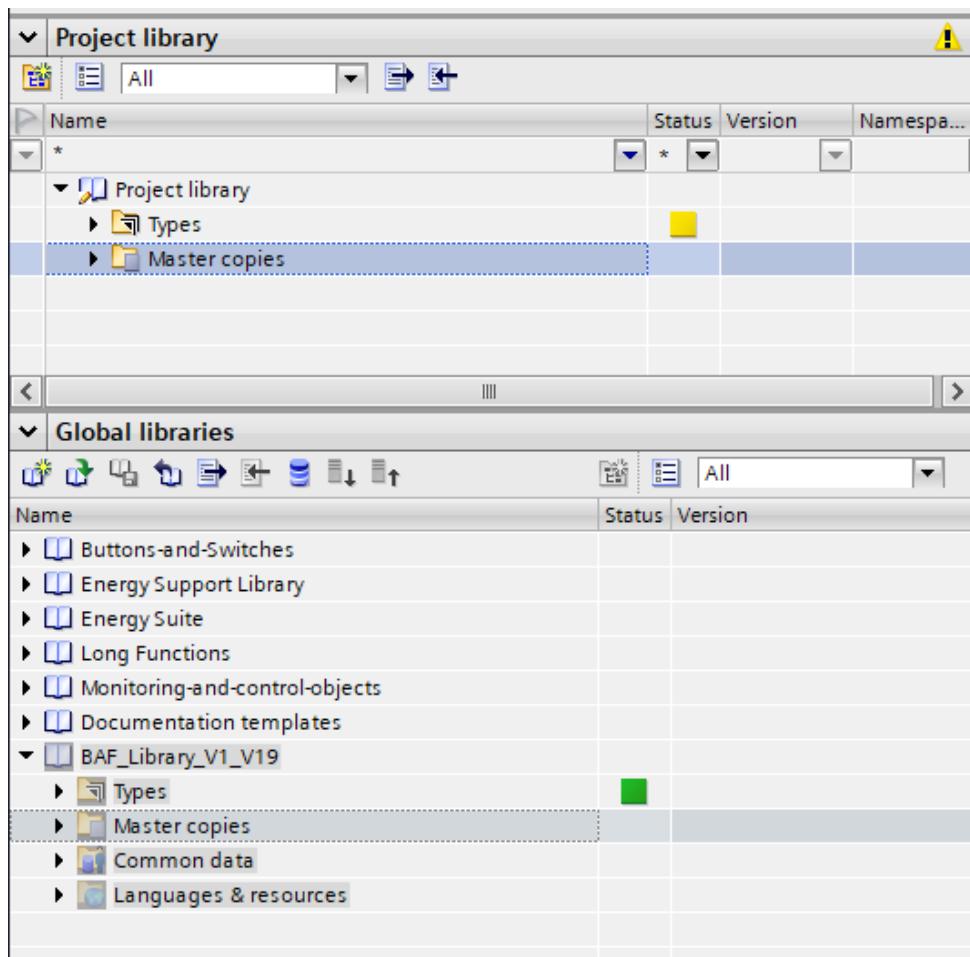


图 4-17: 导入模板副本

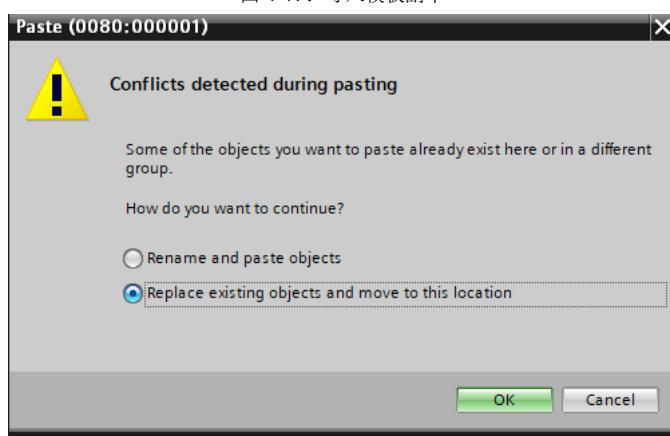


图 4-18 覆盖现有数据

如果出现此信息，请选取“替换现有对象并移至此位置”。

现在，项目库已使用全局库中的模板副本进行更新。

已使用模板副本并在项目中对其进行了修改的最佳实践：

为了避免在实现新的模板副本时在项目中丢失自定义实现（仅当修改了某些内容时），可以使用以下过程作为最佳实践：

- 将新的模板副本拖放到目标设备位置，然后选择“重命名和粘贴对象”选项。

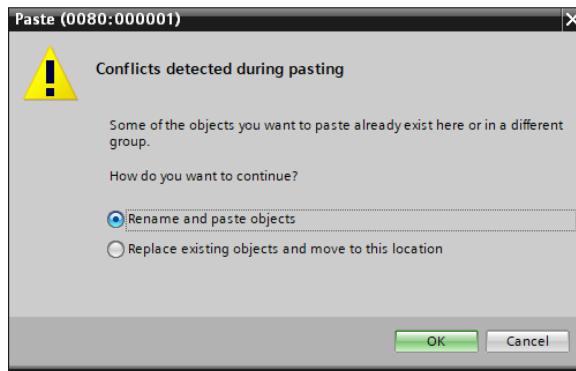


图 4-19 重命名和粘贴对象

结果将如下所示：

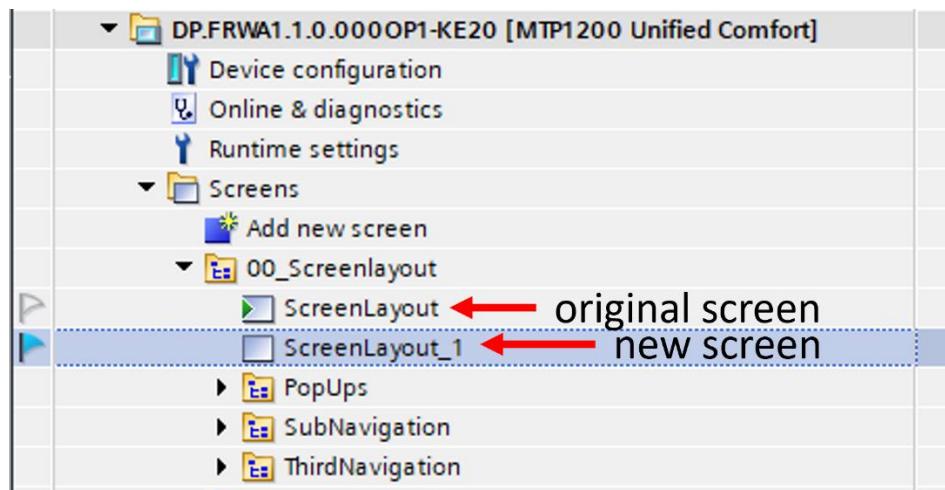


图 4-20 重命名的对象

现在，可以在分屏视图中打开两个屏幕，并根据需要将对象传输到屏幕上。



图 4-21 分屏视图

5. 硬件配置

AF 项目带有一个预组态的应用示例。在本章中，将介绍选定的硬件配置。

5.1. 具有 RT/IRT 组态的 PROFINET IO

PROFINET IO 是一种基于快速以太网的可扩展实时通信系统。提供两种性能级别，PROFINET RT 用于实时关键过程数据，PROFINET IRT 用于高精度和等时同步过程。

在同一个网络中，可将 IRT 通信与 RT 通信进行组合。可以在两个 IO 系统之一中使用 IRT 通信。在较高级别的 IO 系统中或较低级别的 IO 系统中使用 IRT。

AF 项目引入了带有两个 S210 的 IRT 组态和一个带有 IO、HMI 和两个 G120 的 RT 组态。

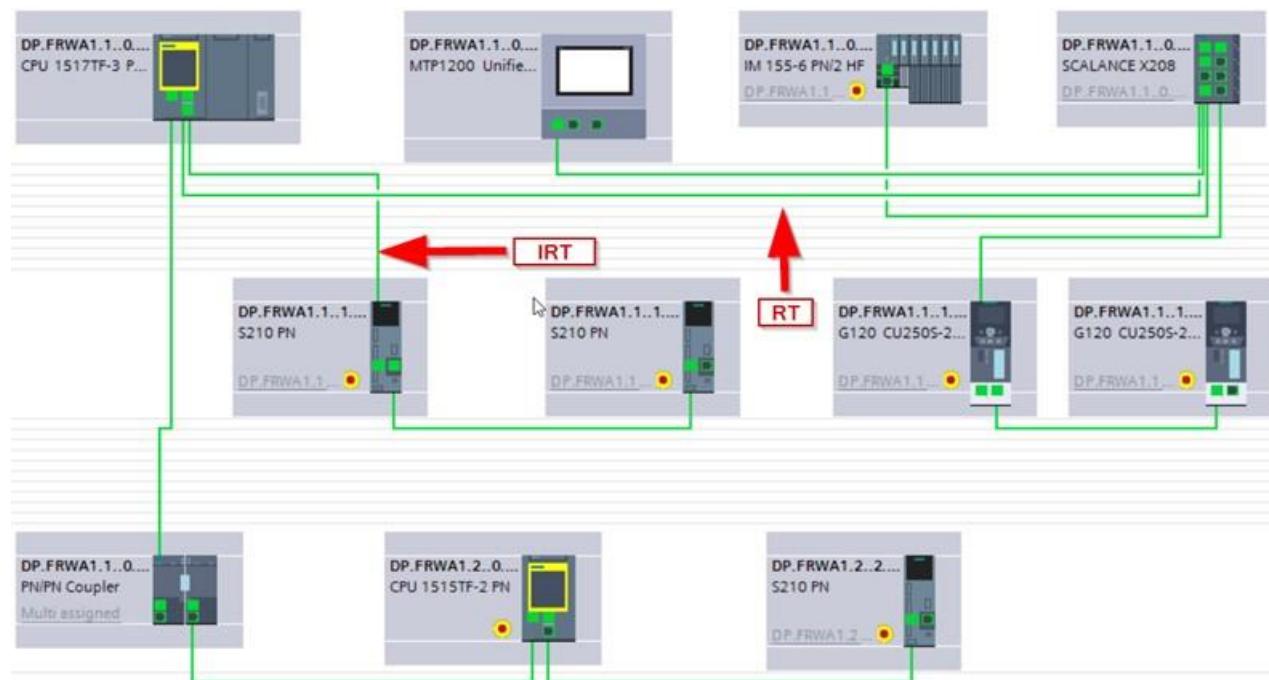


图 5-1 具有 RT 和 IRT 组态的拓扑视图

激活 IRT 后，驱动器组态上有“动态伺服控制”(DSC) 选项。DSC 的优点是将运动控制闭环的一部分从 PLC 转到驱动器上。这大大降低了 CPU 负载，并实现了在驱动器硬件中执行的高速位置控制（见下图中突出显示的内容）。

注

默认情况下，IRT 最多支持 64 个网络设备。如果需要超过 64 个 IRT 设备，可以使用 PROFINET DFP（动态帧打包）和软件控制器 S7-1508S T/TF 来实现。

以下 SIMATIC/SINAMICS 组件支持 DFP（截至 2024 年 6 月）：

- SIMATIC S7-1508S T/TF
- SIMATIC ET 200SP (HS)
- SIMATIC PN/PN 耦合器
- SINAMICS S200
- SINAMICS S210 (6SL5310...)

有关 DFP 的更多信息，请参见功能手册“使用 STEP7 组态 PROFINET”。

[SIOS-ID: 49948856](#)

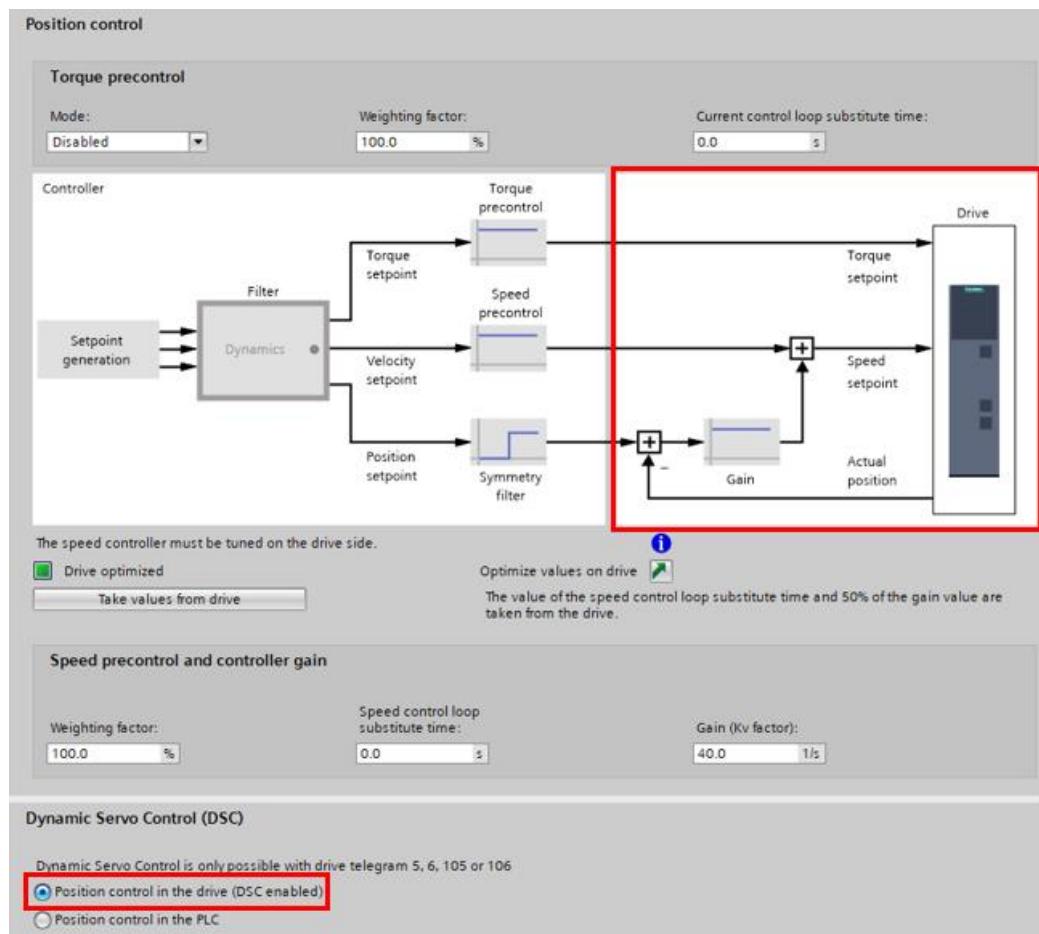


图 5-2 启用了 DSC 的 PROFINET IRT 组态

5.2. PROFINET 拓扑

拓扑组态是 IRT 的先决条件。编程人员必须连接设备之间的特定端口，就像在真实机器中连接的那样。

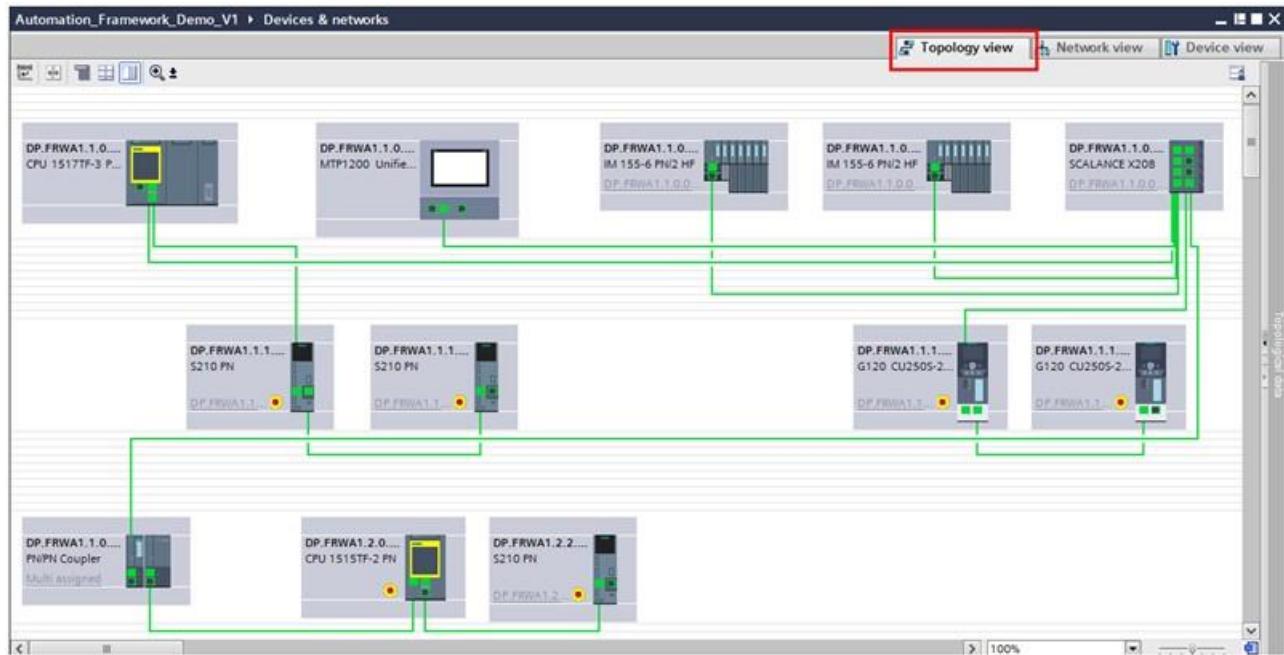


图 5-3 拓扑视图

拓扑信息用于优化数据传输、带宽分配和通信帧，从而保证最佳的 PROFINET 性能。

注

有关 PROFINET 和所有功能的更多信息，请参见
<https://support.industry.siemens.com/cs/ww/en/view/49948856>。

5.3. 时间同步和时间设置

当子系统的日志时间戳与 PLC 日期时间同步时，带有报警日志的系统诊断将受益。有时，PLC 日期时间本身与使用 NTP 网络协议的上级系统同步。以下章节将介绍使用 [LSNTP](#) 库启用级联系统级时间同步所需的设置。

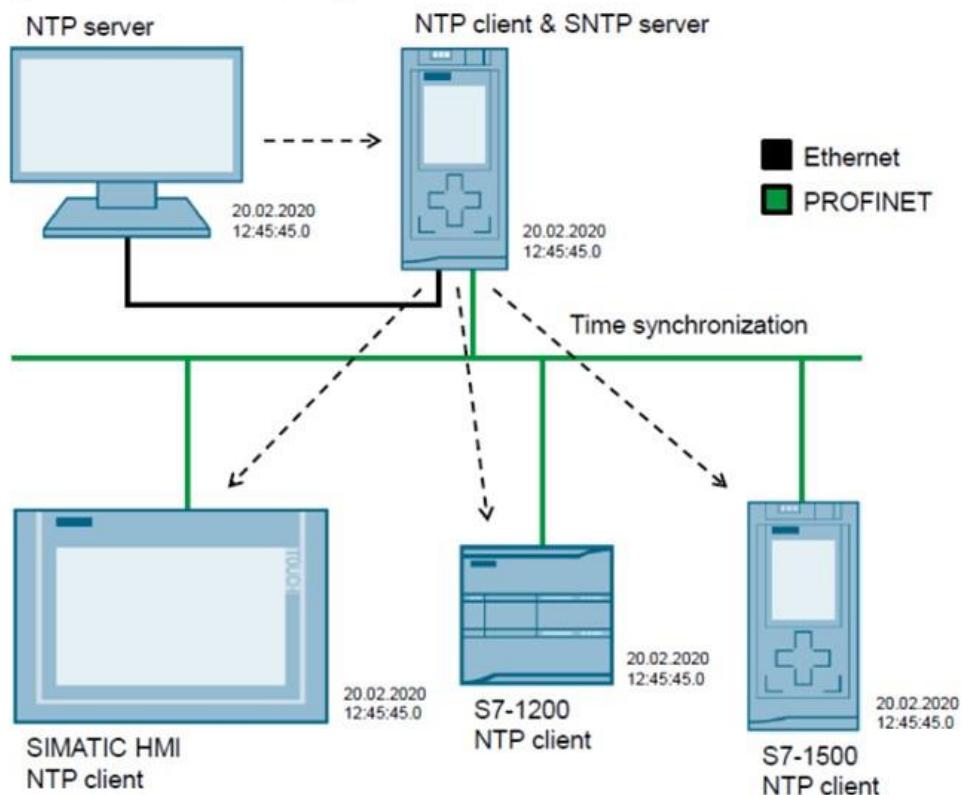


图 5-4 时间同步网络概览

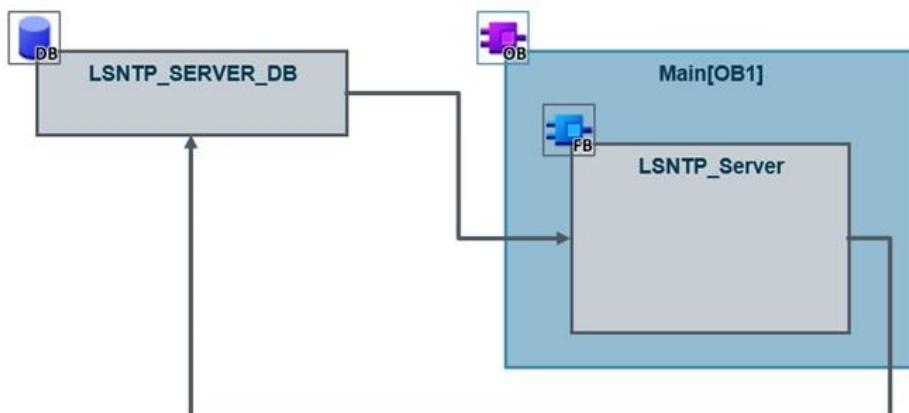


图 5-5 调用结构

技术原理

使用 SIMATIC S7 CPU 作为 SNTP 服务器，可以灵活、简单地同步系统和子系统，例如，获取系统级错误消息和记录数据的时间戳。LSNTP 库提供了一个函数块，用于执行以下功能：

- 接收和评估来自 (S)NTP 服务器的 NTP 报文。
- 创建 NTP 报文并将其发送到客户端以进行时间同步。

SNTP 服务器的误差小于 10 ms。

如果不使用 CPU 作为客户端的高级 NTP，则可通过“设置 - 日期和时间”屏幕设置 CPU 时间。这将通过 Settings.setLocalPlcTime 中的变量将时间和日期发送到 Central Functions - 14_Settings 中的 FC “LAF_SetPlcDateAndTime”。

注

还可以使用外部 NTP-服务器。

VFD (variable frequency drives) 中的 NTP 客户端功能，目前仅受 S120 和 S210 支持。

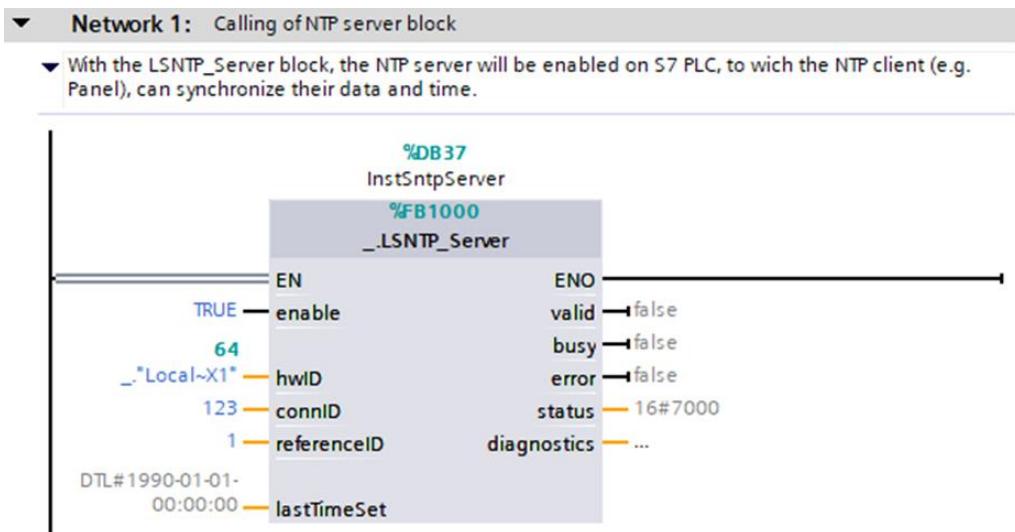
如何使用（请参见软件单元 CentralFunctions）

图 5-6 LSNTP_Server 块激活主 PLC 上的 NTP 服务器，在 OB1 – Main 中调用

- 添加 NTP 服务器 IP 地址（主 PLC）的所有 NTP 客户端都与 NTP 服务器的日期和时间同步。
- 要将 Unified 精智面板组态为 NTP 客户端，必须在 TIA Portal 中组态以下设置。要完成组态，需要下载。

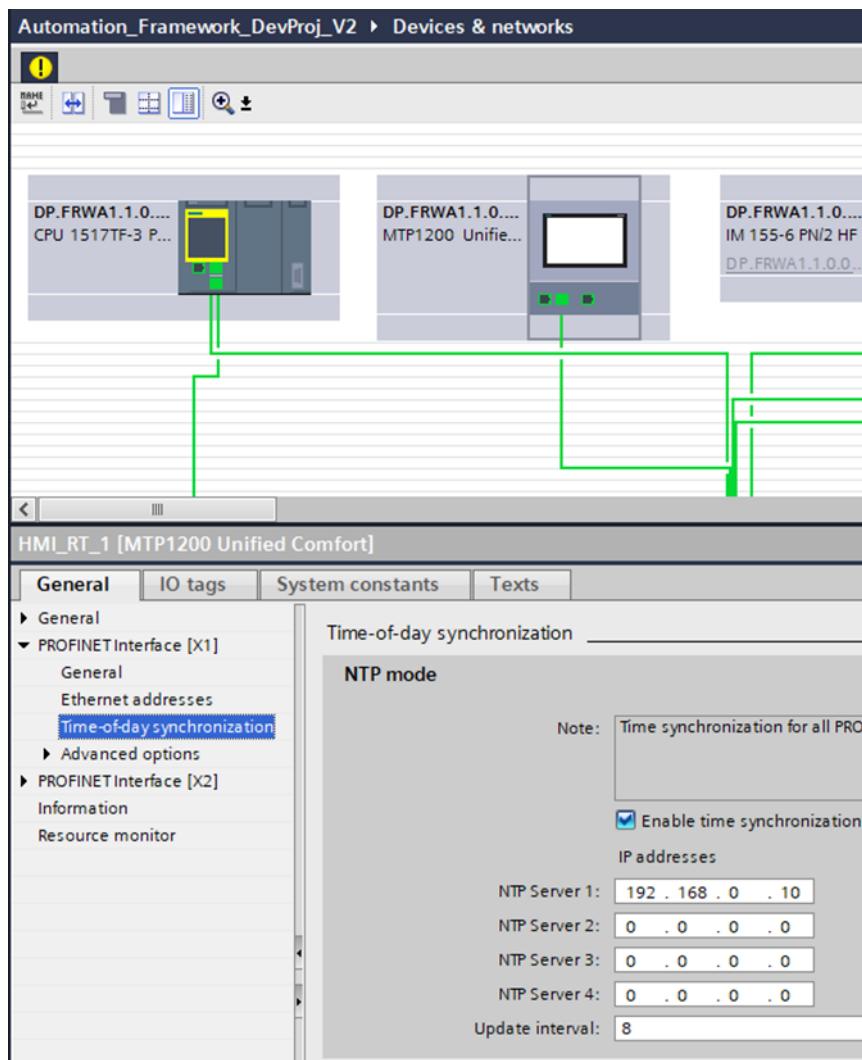


图 5-7 TIA Portal 中的 HMI 设置：初始化 LSntp_Server 块的 IP 地址

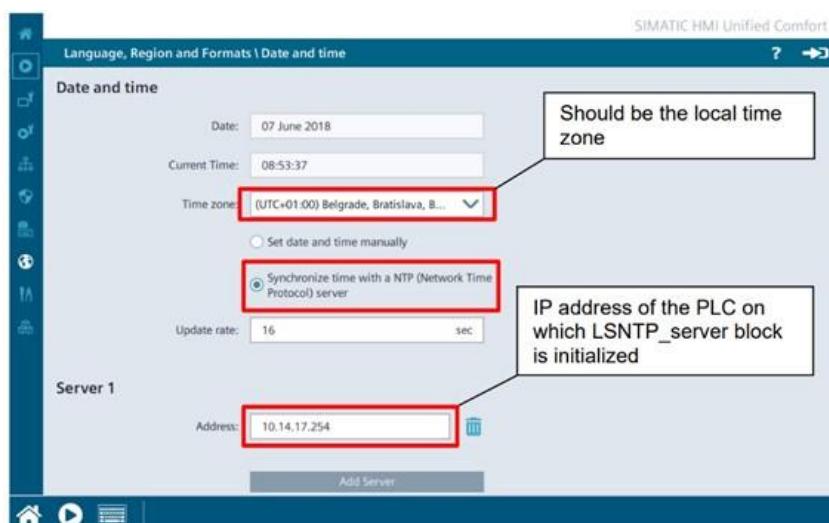


图 5-8 直接在面板上进行 HMI 设置

- 要启用每个 PLC 的 NTP 客户端，必须在 TIA Portal 中组态以下设置。要完成组态，需要下载到 PLC。

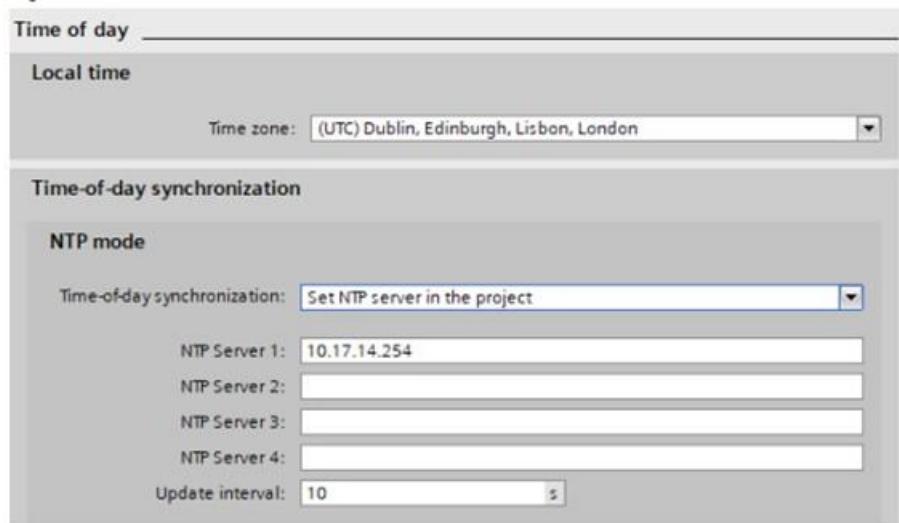


图 5-9 NTP 服务器的组态

- 如果 PLC 时间未通过外部 NTP 服务器同步，则可以通过 AF HMI 手动设置 PLC 时间。

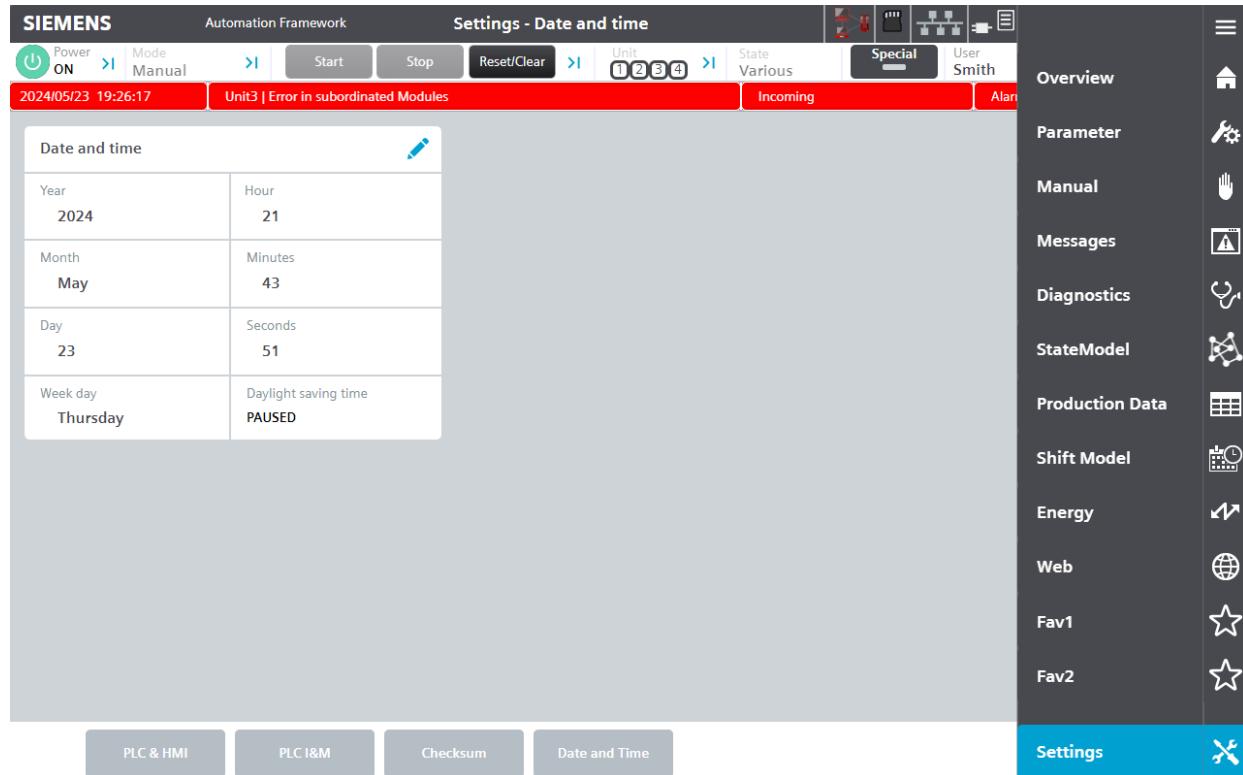


图 5-10 通过“AF 设置”画面手动设置 PLC 时间和日期

6. PLC 软件架构

PLC 软件架构的描述包含有关软件结构的信息，也包含有关程序部件之间通信的信息。

6.1. ISA-88 设计

ISA-88 简介

工业过程可分为连续制造过程、离散制造过程或批量制造过程。连续过程被归类为具有连续流出的过程，例如造纸或钢铁的生产。离散过程具有离散的输出，例如汽车的生产。

批生产过程是一种通过在有限时间内，使用一台或多台设备，将一定数量的输入材料进行有序的加工活动，从而生产出有限数量的物料的过程。

由于以下原因，需要提供一个管理批生产过程的标准：

- 没有通用的批生产控制模型。
- 工厂操作员难以传达加工要求。
- 来自不同供应商的解决方案难以集成。
- 批生产控制解决方案难以组态。

因此，ISA（国际自动化学会）决定建立一个标准，描述一种构建批生产过程的方法。它定义了术语和模型，使批生产工厂的设计和操作更加简单和统一。该标准被命名为 ISA-88，本文档仅引述该标准 (ANSI/ISA-88.00.01-2010) 的第一部分“模型和术语”。

ISA-88 不仅仅是软件、设备或程序的标准。这是一种设计理念。了解 ISA-88 将有助于更好地设计流程和制造产品。

ISA-88 如何发挥作用？

- 模块化可实现更轻松的全局复制和更好的投资回报。
- 实现设备与配方分离。
- 设计概念使验证更容易。
- 符合 ISA-88 的解决方案有助于跟踪产品和过程数据。
- 从客户那里收集需求并将需求传达给供应商的过程将更容易。
- 提供有关如何从异常事件中恢复的准则。

注

有关批生产的更多详细信息，请转到链接的文档。

<https://support.industry.siemens.com/cs/ww/en/view/109784331>

概念

ISA-88 是生产机器模块化编程结构的重要基石。AF 项目引入了一个符合 ISA-88 的物理模型的实现示例，该模型的层级从单元到控制模块 (CM)。

下图显示了这种层级实现。

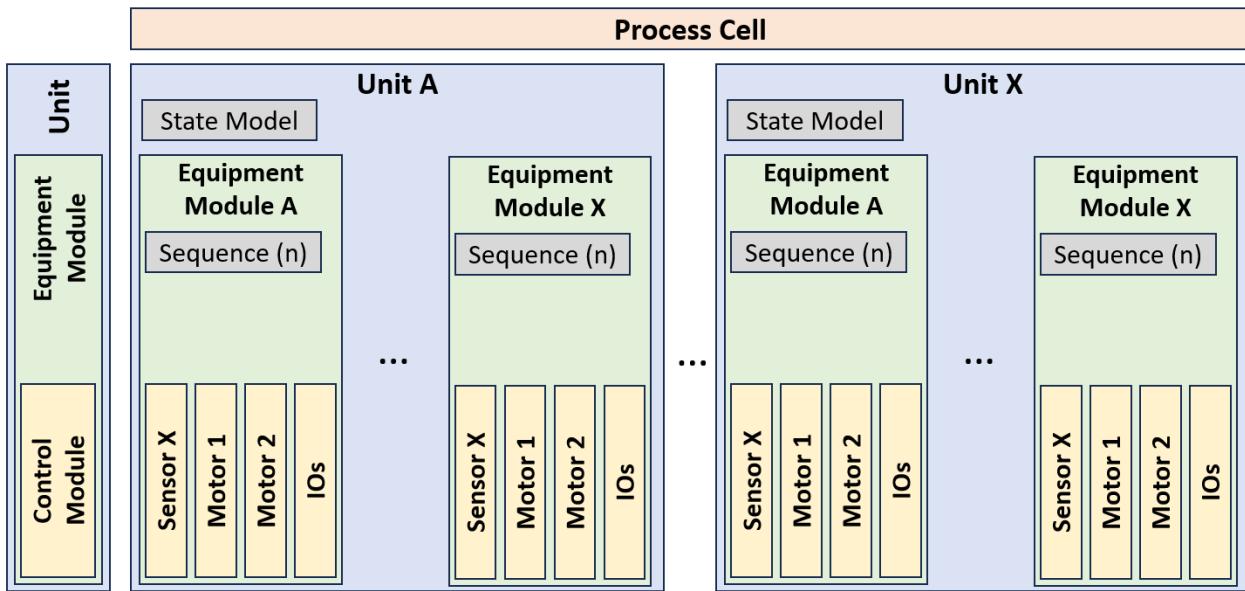


图 6-1 ISA-88 模型

单元分类

一个过程单元包含多个单元。这些单元代表生产中的物理处理设备，例如反应釜。该单元包含设备模块 (EM)，而设备模块 (EM) 又包含控制模块 (CM)。单元是一组独立的设备，通常以主要加工设备（如混合罐或反应釜）为中心，它可以执行一项或多项主要加工活动，如发酵、收获或净化。

通常，组成单元的模块不能由其自身单元以外的设备操作或控制。任何被认为是通用的或由多个单位操纵的设备，都应定义为共享设备，这些设备可以由一个单元临时获取以执行特定任务。

定义批生产单元时，一个重要的考虑因素是单元只能同时操作单个批生产。批生产可以存在于多个单元中，但一个单元在任何给定时间只能容纳一个批生产。

定义单元的边界对于保证工厂的灵活性、模块的可重用性以及多个工厂之间报告的一致性非常重要。工艺流程图 (PFD) 可用于初步识别单元。一旦掌握 P&ID 的详细信息，就可以进一步完善该单元。对智能操作单元进行模块化集成的情况下，单元及其 ISA-88 结构由 OEM 供应商定义。

控制模块 (CM) 分类

一旦定义了单元边界，就可以识别设备模型的下游模块。

最简单的方式是从控制模块 (CM) 开始。这些是物理模型中可以执行基本控制操作（状态导向或监管控制）的最低级别的设备分组，通常与被监视和/或控制的仪器/设备类型有直接关系。例如泵、电机、阀，以及 PID 控制回路。CM 通常直接连接到 IO 信号。CM 还包含与设备、产品和操作员保护相关的异常处理。CM 在 P&ID 图表上很容易识别，如下图所示。控制模块可以由操作员在手动模式下通过 HMI 进行控制，或者在自动模式下由高级 CM 或 EM 进行控制。

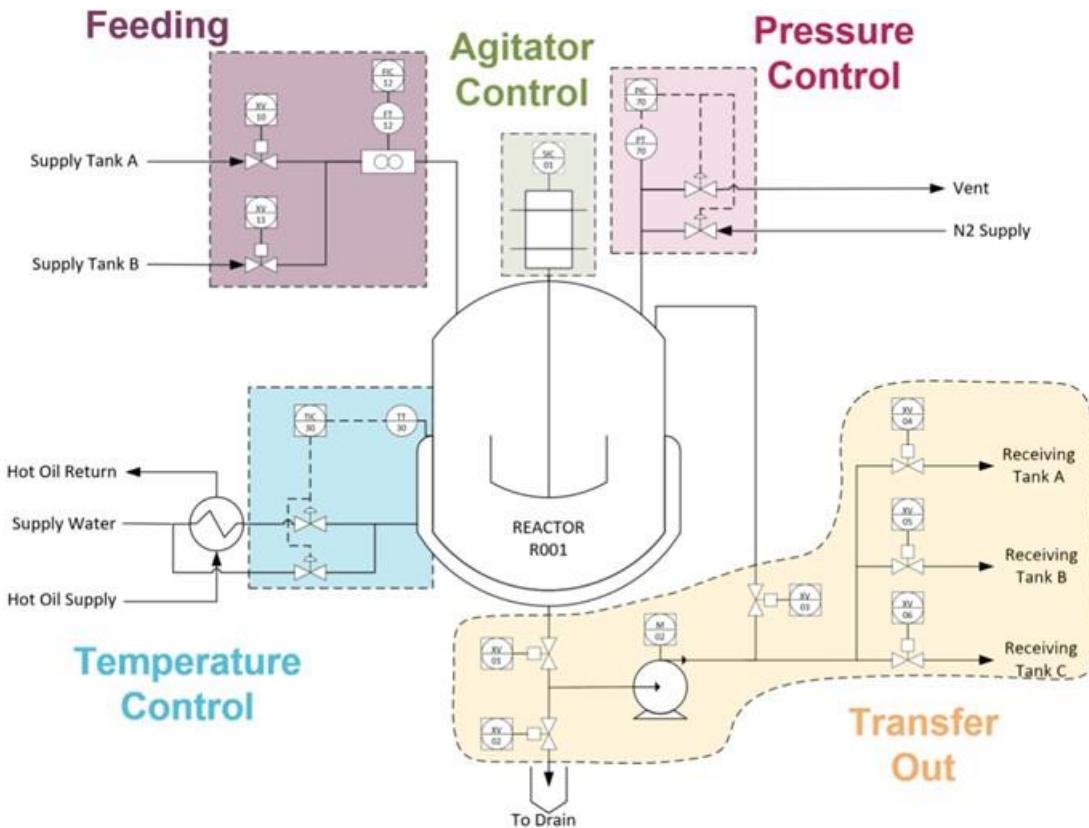


图 6-2 来自工艺单元“反应釜”的示例 P&ID 及其 EM（突出显示）和 CM

设备模块 (EM) 分类

EM 是一组控制模块的功能组，可以执行有限数量的特定次要处理活动，例如定量给料、加料或温度控制。EM 通常以工艺设备为中心，例如容器出口/进口歧管、工艺加热器等。将单元分离为功能性 EM 应该已经反映了过程和程序控制模型中的过程动作和配方阶段。在不改变 EM 基础序列的情况下，应该可以在同一设备上引入新产品。示例反应釜的 EM 如上图所示：进料、温度控制、压力控制、搅拌器控制和输送。

根据 ISA-88，EM 可以包含其他从属 EM，但建议将其限制在没有阶段或批生产要求的区域。

EM 通常是监视和控制所有下属 CM 的序列。EM 还包含异常处理逻辑，用于处理从属 CM 的故障和工艺故障。通常，异常逻辑被标记为保持状态或保持逻辑，并将 EM 驱动到故障安全状态。EM 本身没有在 ISA-88 中定义强制性状态逻辑。EM 通常具有配方感知能力，因此其状态逻辑遵循配方阶段的状态逻辑。

由于 EM 的序列反映了过程操作，因此通常会在整个序列中启用和禁用控制模块相关的报警和过程互锁或更改报警限制。在设计此类命令序列时必须考虑到这一点。

同一单元内 EM 之间共享的控制模块的冲突消除将在 EM 级别执行。

在同一单元中或单元之间共享的 EM 的冲突消除在阶段级别执行。每个阶段都将获得执行特定过程操作所需的设备模块。也可以以相同的方式在另一个单元上获取 EM。

在单元内对 EM 进行分区时，应用以下条件：

- EM 中的所有 CM 都必须支持类似的处理目的，并且是独立的。
- EM 必须是可重用的，以便可以使用“类型-实例”概念。
- 该模块的设计必须使其能够执行所有可用的处理功能，见上文，EM 必须具有在没有其他 EM 交互的情况下自行运行的能力。
- 通过将过程异常限制在 EM 中来最大程度减少过程异常的影响；理想情况下，异常最初只会影响一个 EM，然后会传播到其他 EM 甚至单元。

与 CM 一样，EM 可以由操作员在手动模式下进行控制，也可以在自动模式下由阶段控制。

需要注意的是，理想情况下，CM 和 EM 应遵循模块化方法，其中功能包含在模块中，这些模块可以作为独立模块进行编程和测试，并具有与其他模块的定义接口。为了提高可重用性，在 TIA Portal 中，对控制器和 EM 的类型-实例提供支持。相似的 EM 和 CM 可以归类到一个类型中。可以测试此类型，然后将基于此类型实例化所有模块。

通过创建这些标准的 ISA-88 批生产元素，可以获得巨大的好处：

- 节省软件设计和工程组态的成本
- 在应用实施的初始过程中更快地进行测试和验证
- 提高维护和灵活性 - 通过预先测试的标准类型，可以更快地实现安装的扩展，同时减少（重新）验证工作。

在自动化框架中实现

ISA-88 单元的基本实现如下图所示。一个 AF 单元的逻辑在一个专用软件单元内编程。每个 EM 还有一个专用的软件单元。每个单元都有一个 EMO_General，用于单元级功能，如前提条件、断路器或信号堆栈或气动系统的接口。EMO_General 通常包含非复杂逻辑，不包含序列。此外，其他 EM 与过程相关。总是至少有一个 EM。EM 的最大数量受 PLC 存储器大小的限制。

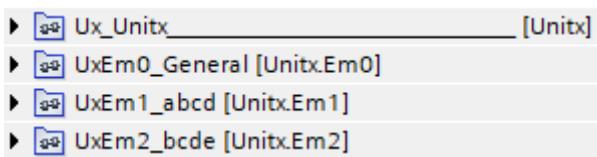


图 6-3 自动化框架中的 ISA-88 单元示例

上述软件单元中的命名是占位符，必须根据实际应用进行调整。建议根据其功能来命名软件单元，而不是根据它们的位置来命名，因为应该允许它们被克隆并在不同的应用中重用。

6.2. 状态机

每个单元都包含自己的状态机。

状态模型

模式和状态完全定义了单元的当前行为和条件。根据模式和状态，设备和控制模块执行各自的任务。在状态模型中，状态以有序的方式排列，与机器的指定操作一致。模式状态管理器根据操作员的命令和过程的反馈确定设备的当前模式和状态。

注 PackML 是 OMAC 的自动化标准，并被 ISA 作为 TR88.00.02 采用，可以更轻松地传输和检索一致的机器数据。PackML 的主要目标是促进整个工厂车间实现一致的“风格和外观”，并促进和鼓励行业创新。OMAC 的 PackML 团队获得全球认可，由控制供应商、OEM、系统集成商、大学和最终用户组成，他们合作制定定义，以符合 ISA-88 标准以及大多数自动化机械的技术和不断变化的需求。

在 AF 中，OMAC PackML 状态机是使用的默认状态机。此状态机可以灵活地重新组态，以满足其他状态机的要求。也可以使用其他状态机库。但是，这将意味着当前 AF 实现的重大变化，如果可能的话，应避免更换状态机。

单元的状态机始终处于定义的模式和状态。模式和状态由两个 DINT 变量表示。各种预定义命令会触发模式或状态转换。此类命令可以由操作员触发（例如通过 HMI 或命令按钮）、来自生产线中的外部系统（例如进料站、在线控制器）或由过程本身触发（例如，由于材料不足或内部错误导致的停止）。该单元的模式状态管理器负责评估所有这些命令，并确定机器当前活动的模式和状态。

AF 中使用的 OMAC PackML 状态机支持以下模式、状态和命令：

Modes		States		Commands	
0	Invalid	0	Undefined	0	Undefined
1	Production	1	Clearing	1	Reset
2	Maintenance	2	Stopped	2	Start
3	Manual	3	Starting	3	Stop
4 .. 15	User definable	4	Idle	4	Hold
		5	Suspended	5	Unhold
		6	Execute	6	Suspend
		7	Stopping	7	Unsuspend
		8	Aborting	8	Abort
		9	Aborted	9	Clear
		10	Holding	10	Complete
		11	Held	11	StateComplete
		12	Unholding		
		13	Suspending		
		14	Unsuspending		
		15	Resetting		
		16	Completing		
		17	Completed		

图 6-4 可在 AF 中使用的模式、状态和命令

单元控制库 (LUC) 提供数据类型 LUC_typeModes、LUC_typeStates 和 LUC_typeStateCommands，以表示模式、状态和命令。这些数据类型在单元及其 EM 内部以及之间使用。

下图显示了 PackML 基本状态模型。它表示已定义状态、状态命令和状态转换的完整集合。对于每种单元模式，都可以根据应用定义这些基本状态的任何集合或子集（参见图 6-6）。

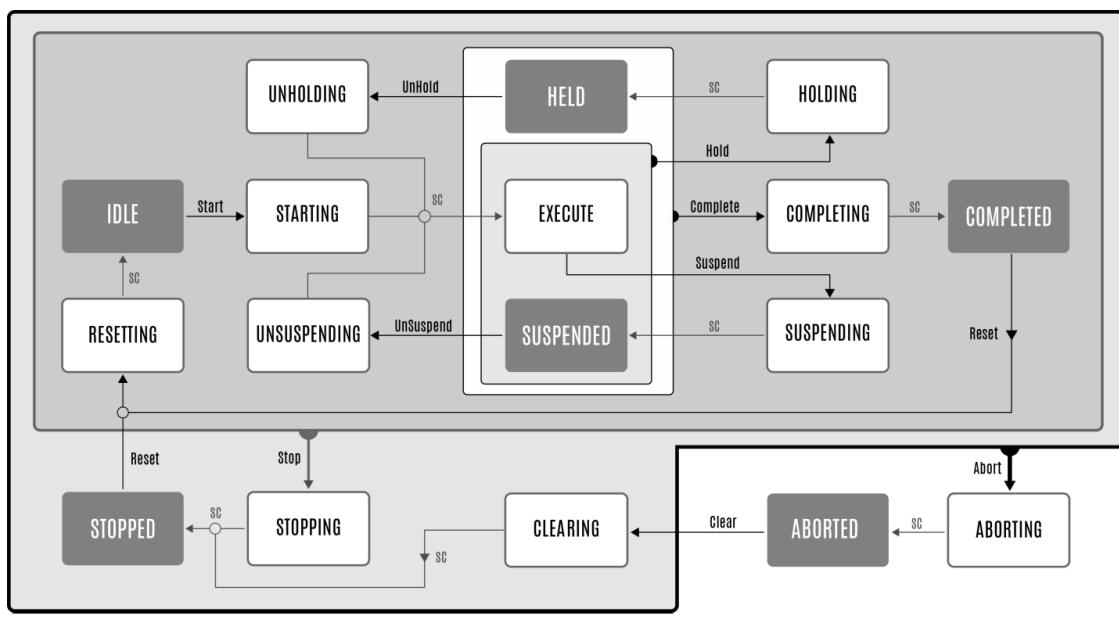


图 6-5 基本状态模型

组态

可以组态状态机。最多可启用 16 种模式。对于每种模式，都可以组态启用哪些状态和转换。

这样一来，状态机就具有极高的定制能力。详细信息在有关单元的章节中介绍。

单元模式

一个单元可以以不同的模式运行，例如生产、维护、手动、清洁、点动模式等。在每种模式下，该单元可以具有不同数量的状态和命令。因此，每种模式的状态机可能看起来各不相同（请参见图 6-6）。

在 AF 中，预组态了以下 OMAC PackML 标准模式：

- **生产（强制）：**

这就是生产常规模式。当选择此模式时，机器会执行相关逻辑，以响应主要来自外部系统或由操作员直接输入的命令。

- **维护（可选）：**

此模式允许授权人员独立于其他系统运行单元/机器。此模式通常用于故障排除、机器试验或测试操作改进。

- **手动（可选）：**

此模式允许直接控制单个机器模块。它可用于单个组件的调试、计划外技术干预的故障诊断等。

可以组态不同或额外的模式（总共多达 16 种模式）。

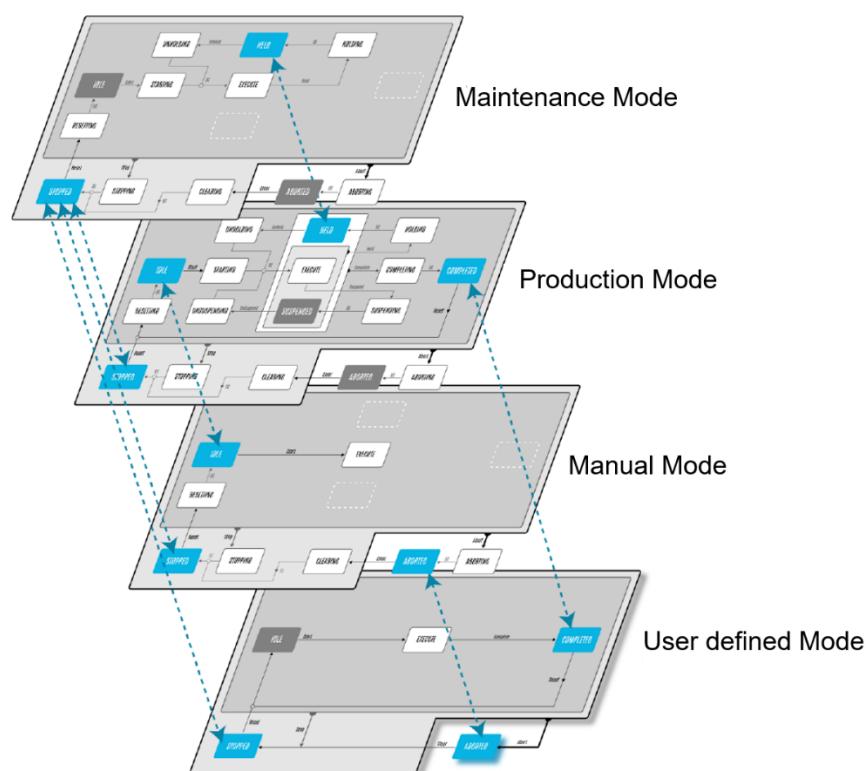


图 6-6 具有不同模式的示例，具有相应的组态状态和允许的模式更改

单元状态

不同的状态可以分为两种类型：

- **激活状态：**表示某些处理活动的状态。这意味着在有限的时间内或直到达到特定条件之前，以逻辑顺序单次或重复执行处理步骤。在 ISA-88.00.01 中，这些被称为瞬态，通常是那些以“ING”结尾的状态，如“STARTING”或“RESETTING”。

- **等待状态:** 用于标识机器已达到定义条件的状态。一旦达到该状态，机器将保持此状态，直到通过内部或外部命令触发某些转换。

定义可能的状态并描述通常应在其中执行的任务：

- **正在清除:** 由状态命令启动，用于清除“正在中止”时可能发生的故障，以及在进入“已停止”状态之前处于“已中止”状态时的故障。
- **已停止:** 完成“正在停止”状态后，机器通电并静止。与其他系统的所有通信都正常运行（如果适用）。“复位”命令将导致从“已停止”退出到“正在复位”状态。
- **正在复位:** 此状态是“已停止”或“完成”状态的“复位”命令的结果。故障和停止原因被复位。“正在复位”通常会导致安全设备通电，并将机器置于“空闲”状态，等待“启动”命令。在此状态下不应发生任何危险运动。
- **空闲:** 此状态表示“正在复位”已完成。机器将保持在“正在复位”状态下达到的条件，并在机器处于“空闲”状态时执行所需的操作。
- **正在启动:** 机器完成启动所需的步骤。此状态是“正在启动”命令（本地或远程）的结果。按照此命令，机器将开始“执行”。
- **执行:** 一旦机器正在处理物料，它就会处于“执行”状态。不同的机器模式将导致特定类型的“执行”活动。例如，如果机器处于“生产”模式，则“执行”将导致正在生产产品，而在“清理”模式下，“执行”状态是指清洁机器的动作。
- **正在停止:** 进入此状态是为了响应“停止”命令。在这种状态下，机器执行逻辑，使其达到受控停止状态，如“已停止”状态所反映的那样。除非已执行“正在复位”，否则无法启动机器的“正常启动”。
- **正在保持:** 当内部（在此单元/机器内部，而不是来自生产线上的另一台机器）机器条件不允许机器继续生产时，使用此状态，即机器由于内部条件而离开“执行”或“已暂停”状态。这通常用于需要少量操作员维修才能继续生产的常规机器状况。这种状态可以自动启动，也可以由操作员启动，并且可以很容易地从中恢复。一个例子是一台机器，它要求操作员定期重新填充胶水分配器或纸盒，并且由于机器的设计，这些操作不能在机器运行时执行。由于这些类型的任务是正常的生产操作，因此不希望执行中止或停止序列，并且由于这些功能是机器不可或缺的一部分，因此它们不被视为“外部”任务。当处于“正在保持”状态时，机器通常会处于受控停止状态，然后在状态完成后转换为“已保持”状态。为了能够在“已保持”状态后正确重新开始生产，在执行“正在保持”程序时，必须将接收到“保持”命令时程序的所有相关过程设定值和返回状态保存在机器控制器中。
- **已保持:** 请参见“正在保持”，了解何时使用此状态。在这种状态下，机器不得生产产品。它要么停止运行，要么继续空转。当内部机器条件发生变化或操作员启动“取消保持”命令时，将转到“正在取消保持”状态。
- **正在取消保持:** 请参见“正在保持”，了解何时使用此状态。当内部条件、物料水平等恢复到可接受的水平时，机器通常会自动进入“正在取消保持”。如果操作员需要执行小规模维护以补充材料或进行调整，则操作员可以启动“取消保持”命令。
- **正在暂停:** 当外部（本单元机器外部，但通常在同一条集成生产线上）工艺条件不允许机器继续生产，即由于生产线上的上游或下游条件，机器离开“执行”时，应使用此状态。这通常是由于堵塞或缺料事件引起的。这种情况可以由本地机器传感器检测到，也可以根据监控系统的外部命令来检测。当处于“正在暂停”状态时，机器通常会处于受控停止状态，然后在状态完成后转换为“已暂停”状态。为了能够在“已暂停”状态后正确重新开始生产，在执行“正在暂停”程序时，必须将接收到“暂停”命令时程序的所有相关过程设定值和返回状态保存在机器控制器中。
- **已暂停:** 请参见“正在暂停”，了解何时使用此状态。在这种状态下，机器不得生产产品。它将停止运行或继续循环而不生产产品，直到外部过程条件恢复正常，此时，“已暂停”状态将转换为“正在取消暂停”状态，通常无需任何操作员干预。
- **正在取消暂停:** 请参见“正在暂停”，了解何时使用此状态。这种状态是过程条件恢复正常的结果。“正在取消暂停”状态将启动将机器从“已暂停”转换回“执行”所需的任何必要操作或序列。为了能够在“已暂停”状态后正确重新开始生产，在执行“正在暂停”程序时，必须将接收到“暂停”命令时程序的所有相关过程设定值和返回状态保存在机器控制器中。

- **正在完成:** 此状态是“执行”状态的自动响应。正常运行即将完成，即进料处的材料处理将停止。
- **已完成:** 机器已完成“正在完成”状态，现在正在等待“复位”命令，然后再转换到“正在复位”状态。
- **正在中止:** 可以随时在响应“中止”命令或发生机器故障时进入“正在中止”状态。中止逻辑将使机器快速安全停止。
- **已中止:** 机器保持与“中止”条件相关的信息。只有在发出明确的“清除”命令后，机器才能退出“已中止”状态，然后进行手动干预以纠正和复位检测到的机器故障。

注

有关此模板中使用的状态机模型的更多信息，请参见：

<https://www.omacl.org/packml>

SIMATIC OMAC PackML V2022 模式和状态管理和机器数据接口¹⁵

6.3. 程序结构

软件单元的使用

自动化框架使用软件单元来构建 PLC 程序。软件单元是一个程序单元，可以单独编译并独立于其他程序单元进行加载。因此，软件单元带来了诸多优势：

- 明确定义的接口和数据处理。
- 独立编译和下载软件各部分。
- 命名空间可以定义，命名约定使用方便。
- 可以使用命名值数据类型 (NVT)。
- 复制和粘贴软件，无需重命名对象。

内容

AF 项目带有预定义的软件单元，用于系统级功能，如 CentralFunctions、Global、Motion、Safety 和程序中使用的项目库中的块。

此外，它还包含四个示例单元，其中 EM 以编程语言 LAD、Graph 和 SCL 进行代码编写。

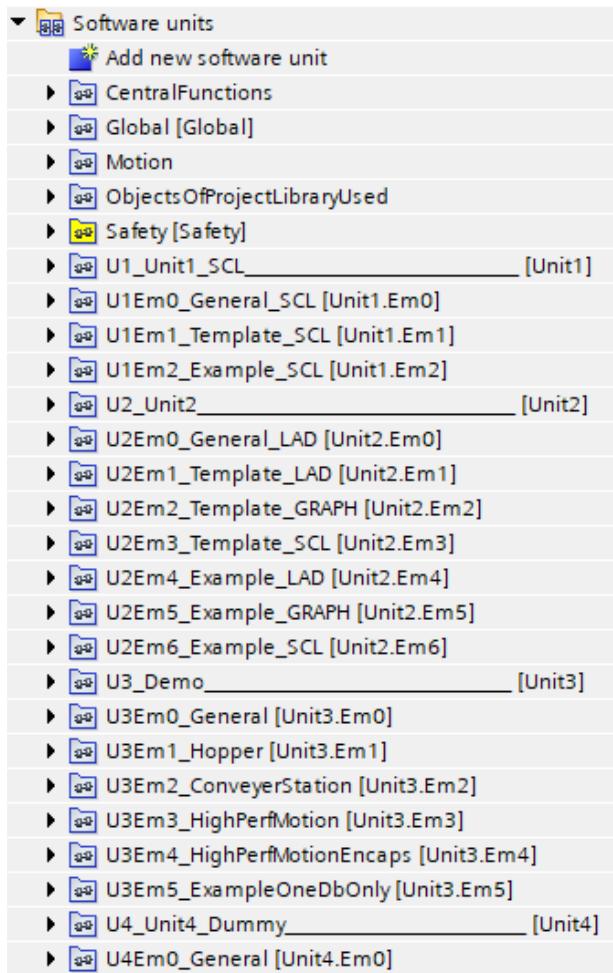


图 6-7 PLC 程序结构

在 AF 项目中，有两种 EM 类型

- **Em_Template:** 仅包含必要的块和调用。它们是开始编写自定义 EM 的基础。每种编程语言对应一个模板。
- **Em_Example:** 包含 EM 的基本示例。每个 Em_Example 包含四个 CM 并实现一个序列示例。所有 Em_Examples 都旨在执行相同的用例，以便用户可以轻松比较不同编程语言中的不同实现。

命名空间

AF 项目中的大多数软件单元都分配了命名空间。相应软件单元中的所有对象都可以与软件单元具有相同的命名空间。使用命名空间，可以克隆整个软件单元，而不会发生命名冲突。命名空间和程序元素的名称共同产生一个唯一的名称，通过该名称可以在 CPU 中识别程序元素。这样，如果不同的程序元素位于不同的命名空间中，则可以对它们使用相同的名称。

AF 项目中的三个软件单元没有分配命名空间，因为它们在整个 PLC 程序中应保持唯一。由于 TIA Portal 工程组态限制，安全软件单元必须是唯一的。这些软件单元包含：

- CentralFunctions
- Motion
- ObjectsOfProjectLibraryUsed
- Safety

软件单元概述和关系

默认情况下，不同软件单元中的程序是隔离的，无法相互交互。但是，在某些项目中，软件单元之间的交互可能是必要的，可以通过 DB 进行数据交换，也可以调用函数块 (FB) 或函数调用 (FC)。为此，用户必须与软件单元建立“关系”，才能访

问其元素，例如 FB、FC、数据库、变量表或用户自定义类型 (UDT)。这些元素一旦公开，就可以访问。应根据需要设置与目标对象 (TO) 的关系，但 AF 项目的运动软件单元除外，其中包括 TIA Portal 运动 OB。

下图显示了 AF 中软件单元和已发布块的关系。

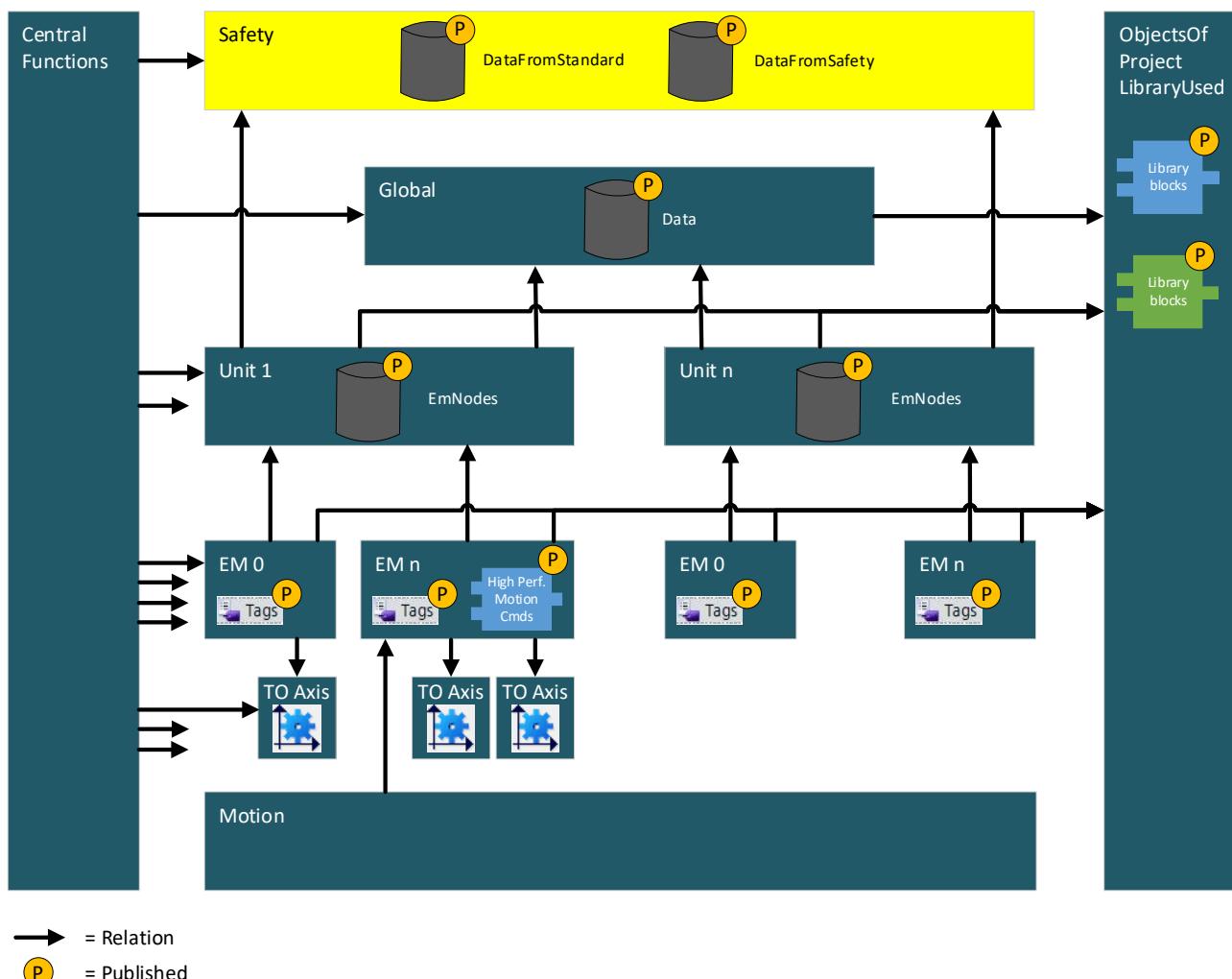


图 6-8 软件单元之间的关系

软件单元描述

- CentralFunctions:** 包括系统级功能和诊断功能。可以与所有其他软件单元建立关系。然而，没有其他软件单元可以与 CentralFunctions 有关系。其他软件单元需要与 CentralFunctions 共享的所有数据都需要由特定的软件单元通过 CentralFunction 访问的 DB 提供。否则，将存在不允许的关系回路。
- 全局:** 包括所有单元的全局数据，不包含任何逻辑，它是一个纯粹的被动软件单元。它与 ObjectsOfProjectLibraryUsed 有关系，因为它使用库中的 PLC 数据类型。
- 运动:** 包括运动 OB。它们执行运动工艺对象，如轴或凸轮输出。每个 Motion OB 在整个 PLC 中只能存在一次，因此所有单元的所有轴都同时执行。运动软件单元不需要与 TO 存在关系，由 PLC 在内部处理。

可以通过两种不同的方式调用运动命令：

- 在标准 EM 循环 (EM 的 OB Main) 中。这样一来，可能会经过多个运动伺服周期，直到 OB Main 调用新的运动命令并将其应用于轴。
- 在运动 OB 中（例如 OB MC_PreInterpolator）。这样，在伺服周期中调用新命令，在下一个伺服周期中立即将新命令应用于轴。对于需要对新位置命令实现最高反应时间的应用，这可能是必要的。有关更多信息，请参见“运动控制”一章，并在软件单元 HighPerformanceMotion 中实现了最佳实践（参见 [EM 演示实例 - 高性能运动](#) 一章）。

只有当必须在某个伺服周期内执行 EM 中的运动命令时，才需要运动软件单元到 EM 的关系。对于所有其他非时间关键运动命令，不需要这种关系。

- **ObjectsOfProjectLibraryUsed:** PLC 程序中使用的库的本地实例。它与其他软件单元没有任何关系。它只包含其他软件单元使用的块。
- **安全:** PLC 中只允许有一个安全软件单元。PLC 的所有安全功能都必须在此软件单元内进行编程。安全软件单元与其他软件单元没有任何关系，完全独立。它不会主动在其他位置写入信息，只提供信息并接收来自其他软件单元的命令。

注

安全软件单元不能由用户创建，但如果添加了新的 PLC 并在添加新的 PLC 之前进行了相应的设置，则由 TIA Portal 自动创建。

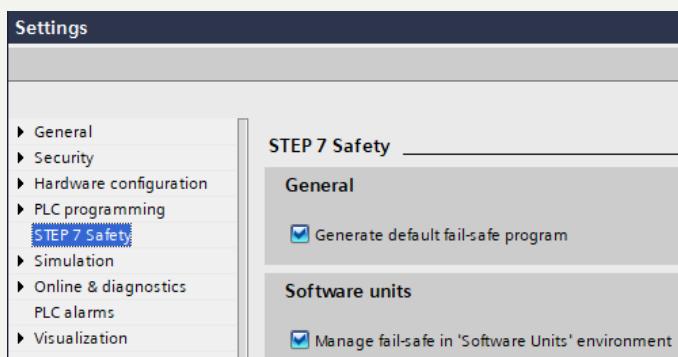


图 6-9 用于在软件单元中管理安全程序的 TIA Portal 设置

- **单元:** 每个 ISA-88 单元都在一个专用的软件单元中实现。它们应与以下各项有关系：

- 全局：存储数据和读取跨单元命令
- 安全：监控安全状态并触发安全命令
- **ObjectsOfProjectLibraryUsed:** 使用库中的元素

- **设备模块:** 每个 EM 都应在专用软件单元中实现。它们应与以下各项有关系：

- 它的单元：监控设备状态并写入状态反馈
- **ObjectsOfProjectLibraryUsed:** 使用库中的元素
- TO：仅在需要时，因为 TO 轴不能嵌入到软件单元中。执行 MC 命令的 EM 需要通过关系访问 TO。

编程语言

AF 项目提供了以不同编程语言实现的各种单元和 EM 模板和示例。用户可以根据自己的编程偏好选择从哪个实现中复制到他的应用程序。

软件单元	描述
U1_Unit1_SCL	完全在 SCL 中编程
U1Em0_General_SCL	完全在 SCL 中编程
U1Em0_Template_SCL	完全在 SCL 中编程
U1Em1Example_SCL	完全在 SCL 中编程
U2_Unit2	在 LAD 中编程
U2Em0_General	在 LAD 中编程
U2Em1_Template_LAD	通用逻辑 + 在 LAD 中编程的顺控
U2Em2_Template_GRAPH	用 LAD 编程的通用逻辑，用 GRAPH 编程的顺控

软件单元	描述
U2Em3_Template_SCL	用 LAD 编程的通用逻辑, 用 SCL 编程的顺控
U2Em4_Example_LAD	通用逻辑 + 在 LAD 中编程的顺控
U2Em4_Example_GRAPH	用 LAD 编程的通用逻辑, 用 GRAPH 编程的顺控
U2Em4_Example_SCL	用 LAD 编程的通用逻辑, 用 SCL 编程的顺控
U3_Demo	在 LAD 中编程的单元示例
U3Em0_General	在 LAD 中编程的通用 EM 示例
U3Em1_Hopper	技术模块的实现示例
U3Em2_ConveyorStation	使用 GRAPH 实现的 EM 示例
U3Em3_HighPerfMotion	如何在标准周期和运动周期中拆分运动调用的示例
U3Em4_HighPerfMotionEncaps	完全封装的 EM 的示例。数据切换完成后, 才能实例化。通过接口进行数据切换。
U3Em4_ExampleOneDbOnly	用一个 DB Data 替换 3 个 DB HmiInterface、DB ControlNodes 和 DB Config 的示例
U4_Unit4_Dummy	LAD 中的模板单元
U4Em0_General	在 LAD 中编程的通用 EM 模板

表 6-1 使用的编程语言概述

6.4. 程序执行

在本章中, 将介绍单元和 EM 内部的默认 Main OB 调用结构和接口处理。目标是确保程序结构最大限度实现模块化, 以便可以轻松克隆软件单元。

每个软件单元都有一个专用的 OB Main, 但 Global 和 ObjectsOfProjectLibraryUsed 除外。AF 遵循以下调用结构:

- 在一个单元内, OB Main 调用 FB CallUnit, 并通过 FB 接口传递来自连接的软件单元的所有外部数据及其公共数据。
- 在 EM 中, OB Main 调用 FB CallEquipment, 并通过 FB 接口传递来自连接的软件单元的所有外部数据及其公共数据。

任何编程逻辑都将在 FB CallUnit / FB CallEquipment 内部执行。通过这种方式, 编程逻辑与对其他外部软件单元的依赖关系分离。在复制和粘贴整个软件单元时, 只需更新软件单元之间的关系。软件单元内部的名称和数据访问权限保持不变。由于使用了命名空间, 因此名称也不会更改。

以下原理图显示了两个软件单元的示例。SWUnit_2 与 SWUnit_1 有关系。这样, 它就可以访问 SWUnit_1 的已发布 DB 的数据。SWUnit_1.PublishedDB 的数据在 SWUnit_2 连接到 FB CallUnit 的 InOut 接口。两个 FB CallUnit 内部的代码现在可以在内部访问共享数据, 而无需直接依赖地址。有关详细信息, 请参见[软件设计原则](#)一章。

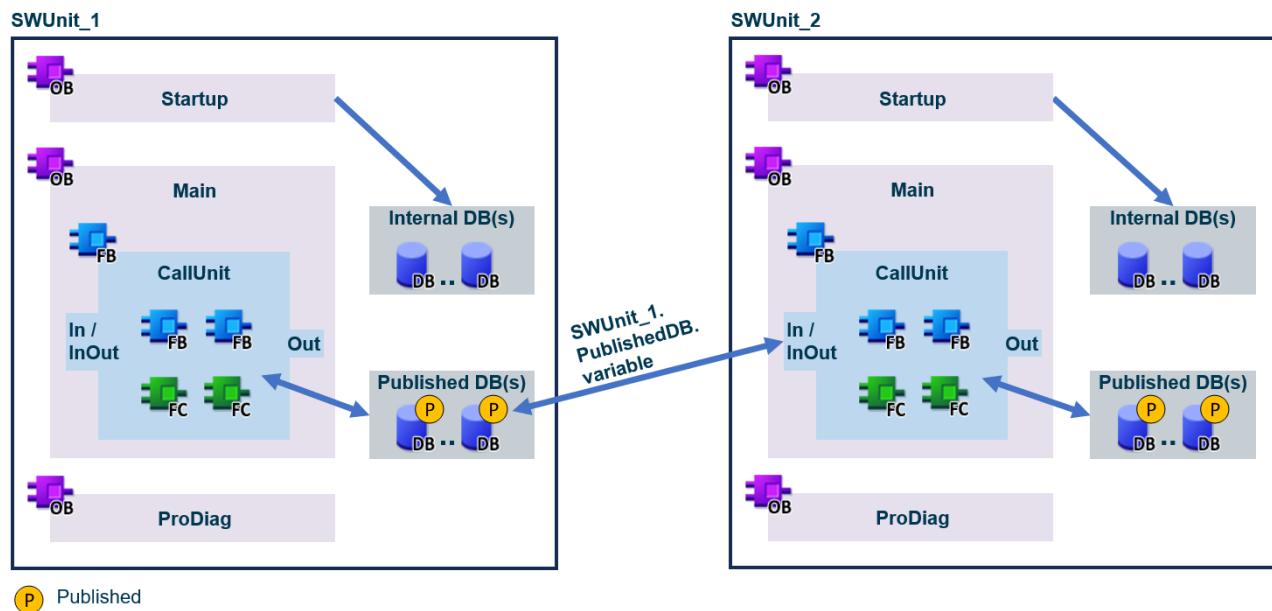


图 6-10 两个软件单元之间的关系以及内部和外部的数据访问

下图展示了一个单元中，在主组织块（OB）“Main”中调用功能块（FB）“CallUnit”的情况。可以看出，来自自身软件单元之外的数据，例如来自“Global”（全局）或“Safety”（安全）的数据，是通过功能块“CallUnit”的接口进行传递的。而自身软件单元内部的数据则不通过块接口传递，而是直接在块内部通过使用直接数据块访问进行处理。这样做带来的好处是，可以实现交叉引用，并且用户可以通过单击直接跳转到定义处。

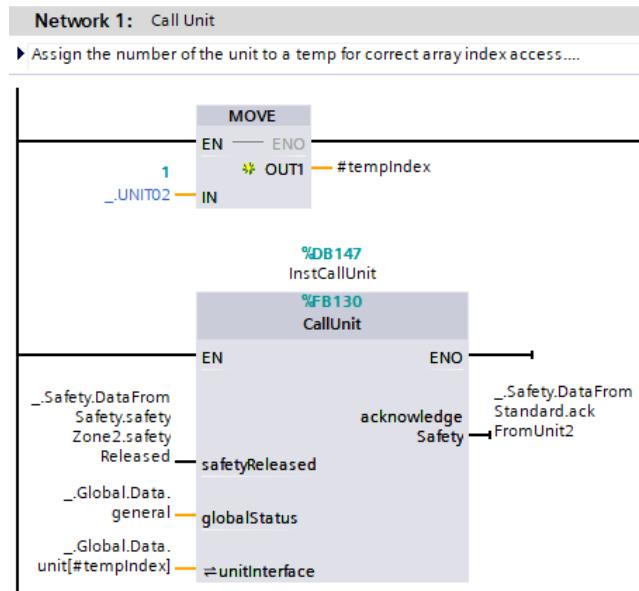


图 6-11 单元的 OB Main 示例

注

CallUnit 的实例数据块没有按照编程风格指南命名（应为 InstCallUnit），而是保留了默认名称（CallUnit_DB）。这是因为在实例中使用“初始值获取”功能时，需要遵守 ProDiag 规则。更多详细信息，请参阅 S7-1500 的 TIA Portal 帮助“获取初始值”一章。

每个软件单元通常包含以下块：

- **OB Startup:** 这些 OB 在 PLC 从 STOP 切换到 RUN 后一开始就被调用一次。常规分配、组态设置或默认值定义通常在这里进行编程。启动例程完成后，操作系统将读取过程映像输入并启动循环程序。
- **OB Main:** PLC 的主要任务。它在循环回路中运行，仅被其他高优先级 OB 中断。它调用用户编程的逻辑。
- **OB ProDiag:** 也像 OB Main 一样在循环中执行。它调用 FB ProDiag，它评估所有组态的 ProDiag 监控并触发报警。
- **内部 DB:** 例如，内部 DB 可以是用于组态数据的 DB Config，也可以是用于与 HMI 共享的数据的 DB HmiInterface。这些 DB 包含的数据仅限于自己的软件单元，并且不能共享。
- **发布的 DB:** 这些数据块用于与其他软件单元共享的数据。它们被用作软件单元之间的连接或全局数据集/中心。
- **用户自定义 FB/FC:** 可以对单个逻辑的用户自定义块进行编程。
- **库块的调用:** 每个软件单元都与 ObjectsOfProjectLibrary 有关系，并且可以从那里实例化块。

6.5. PLC 启动和循环调用序列

用户程序的执行顺序取决于初始事件（PLC 启动或循环操作）、OB 优先级和 OB 编号。软件单元对调用序列没有影响。

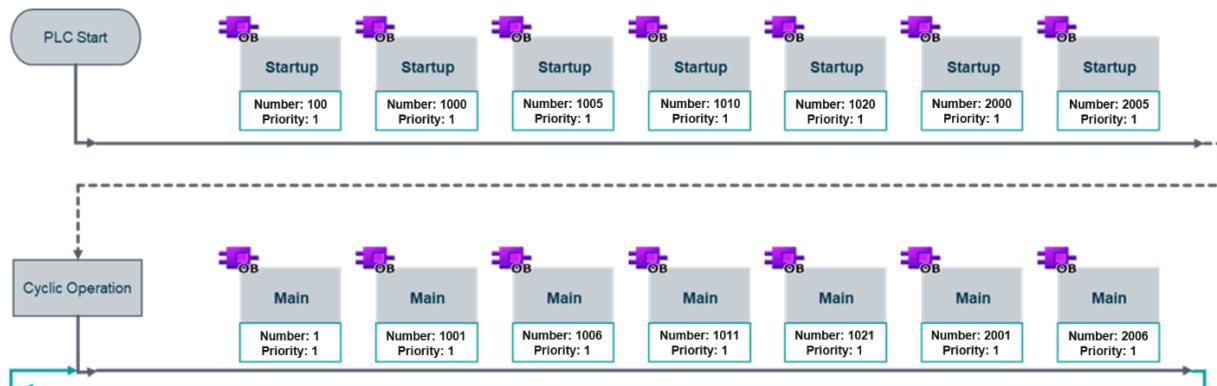


图 6-12 启动 OB 和循环 OB 的调用层级结构

用户程序的基础是循环程序的执行。当执行所有程序循环 OB、OB Main 时，PLC 的操作系统以相同的顺序从头开始重新开始执行。这些 OB Main 具有最低优先级 1。循环程序通常会在任何时候被具有更高优先级的另一个事件类的事件打断。对于多个程序循环 OB，它们按照其 OB 编号的顺序依次调用。首先调用具有最低 OB 编号的程序循环 OB。

以下事件将启动循环程序：

- PLC 启动处理结束。
- 结束对循环程序的上一次运行的处理。

循环程序完成后，操作系统将按如下方式更新过程映像：

- 它将过程映像输出的值写入输出模块。
- 它读取输入模块的输入，并将这些输入传输到过程映像输入。

OB 编号规则

在 AF 中，每个带有用户代码的软件单元都有自己的 OB Main、OB Startup 和 OB ProDiag。为确保正确的调用层级结构，将使用以下编号约定：

OB 编号 = uxxo，其中

u: 单元编号从 1-9

xx: EM 编号从 0-99

o: 0 = 启动 OB, 1 = Main OB, 2 = ProDiag OB, 3/4 = 空, 5 = StartupEM0, 6 = MainEM0, 7 = ProDiagEM0, 8/9 = 空

示例:

OB 编号 = 2001: 单元 2, Main OB

OB 编号 = 3022: 单元 3, EM 2, ProDiag OB

单元序号	单元启动	单元Main	单元ProDiag	EM 0启动	EM 0Main	EM 0ProDiag	EM 1启动	EM 1Main	EM 1ProDiag
1	1000	1001	1002	1005	1006	1007	1010	1011	1012
2	2000	2001	2002	2005	2006	2007	2010	2011	2012
3	3000	3001	3002	3005	3006	3007	3010	3011	3012
4	4000	4001	4002	4005	4006	4007	4010	4011	4012

表 6-2 OB 编号摘录

与 AF 项目中的单元或 EM 无关的其他 OB 具有以下 OB 编号:

名称	OB	#
CentralFunctions	启动	0100
CentralFunctions	Main	0001
CentralFunctions	ProDiag	0250
安全	ProDiag	0251
全局	ProDiag	0252

表 6-3 上位函数 OB 编号

使用此 OB 编号约定, 程序将在循环操作期间按以下顺序执行:

1. 中央功能 OB Main
2. 中央功能 OB ProDiag
3. 安全 OB ProDiag
4. 全局 OB ProDiag
5. 单元 1 OB Main
6. 单元 1 OB ProDiag
7. 单元 1, EM 0 OB Main
8. 单元 1, EM 0 OB ProDiag
9. 单元 1, EM 1 OB Main
10. 单元 1, EM 1 OB ProDiag
11. ...
12. 单元 2 OB Main
13. 单元 2 OB ProDiag
14. 单元 2, EM 0 OB Main
15. 单元 2, EM 0 OB ProDiag
16. 单元 2, EM 1 OB Main
17. 单元 2, EM 1 OB ProDiag
18. ...

具有较高优先级 (> 1) 的 OB 将中断循环程序执行的顺序, 例如 OB 安全、OB 伺服或中断 OB。

6.6. 命令和数据流

本章介绍了单元控制命令（启动、停止、中止等）的处理以及通过 AF 项目和用户代码进行命令传播。

单元命令流传播

自动化框架提供了通过多个 HMI 控制多个单元的架构。因此，单元命令流从指定的 HMI 开始，命令通过 LAF_HmiHeaderManager 转发到全局软件单元的中央数据存储 DB 数据。

然后，每个单元根据当前单元的状态和模式评估来自 HMI 或外部通过 OPC UA 传入的控制命令。传入的命令通常会触发到新的单元状态或单元模式的转换，但并非所有命令都始终被允许。然后，将单元的当前状态和模式发布在 DB EmNodes 和 DB 数据中。

每个 EM 都从其单元中读取状态，并相应地执行其编程逻辑。每个 EM 都必须将其状态反馈给单元。这可能会触发单元状态更改，例如，由于 EM 或从属 CM 内部的故障或某些 CM 内部的硬件按钮。来自 EM 的反馈将被写入单元的 DB EmNodes。EM 反馈是强制性的，除非反馈中的 isLinked 标志为 false。未关联的 EM 可以独立于单元状态进行操作，例如在调试或服务期间通过 em.directModeStateControl。有关 EM 连接工作原理的详细信息，参见[与单元链接](#)一章。

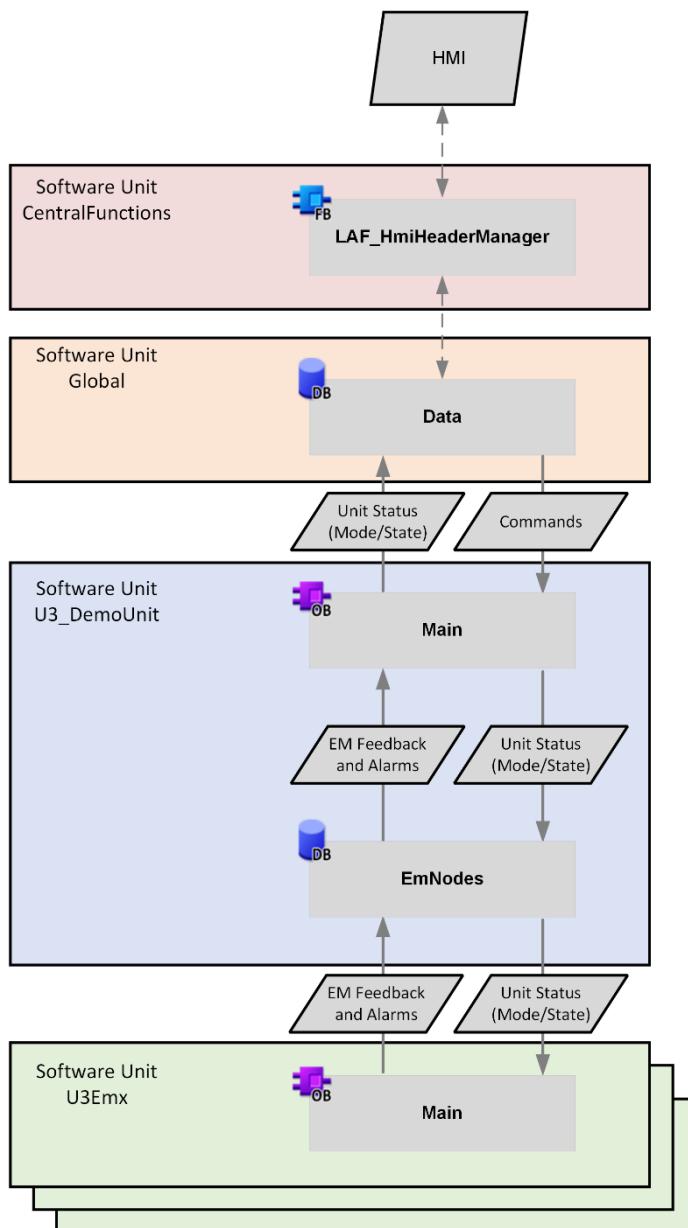


图 6-13 命令和数据流

HMI 接口

AF HMI 通过 Global 和 CentralFunctions 软件单元与单元进行交互。“CallHmi”块在 CentralFunctions 软件单元的 OB Main 中被调用，用于处理全局软件单元内的 DB 数据信号。从这一点开始，与单元的 LUC_InterfaceManager 之间的数据访问是双向的。

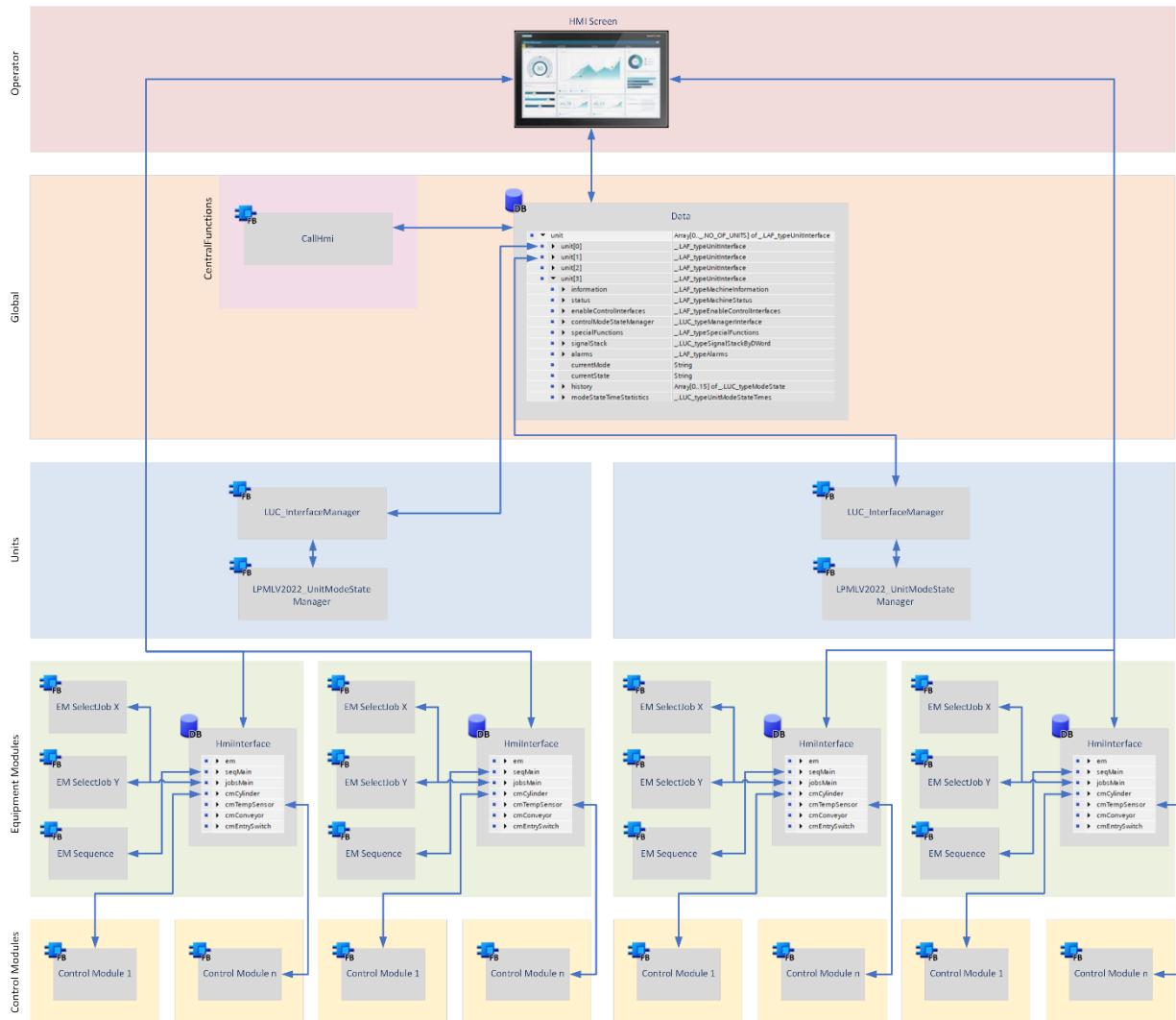


图 6-14 使用 DB 接口和 HMI 连接

此外，每个 EM 都包含一个 DB HMIInterface，其中包含 EM 逻辑及其子 CM 的可视化和操作所需的所有数据结构。HMI 操作的示例包括手动控制 EM 序列执行和/或 CM 的面板。

6.7. 软件设计原则

6.7.1. 通过块接口传递数据与直接全局访问

编程模式

在 PLC 软件设计中，常见的有两种实现模式：

- 在块层级的最高级别连接全局数据（输入、输出、数据块），并且仅使用块接口将信号传递到较低级别
- 在不使用块接口的情况下，直接从源连接较低级别的全局数据（输入、输出、数据块）

两种模式各有优点和缺点：

模式 1（使用块接口）：

优点	缺点
函数和函数块的可重用性更好，因为它们是完全封装的。	交叉引用的功能有限。
可以完全进行版本控制。	将变量从一个块拖放到另一个块时（例如，将控制节点变量拖放到序列中）时，并不是很方便。
	程序员在调试程序和调试设备期间的可用性降低

表 6-4：使用专用块接口切换数据的优点和缺点

模式 2（使用对输入块和数据块的直接访问）：

优点	缺点
通过“转到定义”工具提示，可以轻松跳转到确切的数据源。无需通过接口进行追踪。	当复制块并使用其他数据再次调用时，可能需要重新连接块内的变量调用，从而降低可重用性。
变量拖放操作方便，因为在彼此相邻的拆分编辑器中打开单独的块。	增加了覆盖变量的风险，因为它们可以从不同的位置轻松写入。

表 6-5：直接从每个块读取和写入全局数据的优点和缺点。

总之，模式 1 可实现更高的标准化。通过这种方式，可以更好地实现自动项目生成以及可重用性更高的程序。它被认为是更高层级的良好编程。但是，程序员在调试程序和调试设备期间的可用性降低。

另一方面，模式 2 是一种更基础的编程风格，更易于理解和调试，但在调整新程序时需要付出更大的努力。

使用软件单元简化了模式 2 的实现，因为现在可以复制整个软件单元，而无需重新连接所有内部变量。

在 AF 中实现

用户可以按照一种或另一种模式使用 AF 库进行编程。AF 项目根据模式 2 进行编程，以使 AF 框架产品尽可能通用和简单。特别是，在 TIA Portal 中使用交叉引用功能的可能性大大提高了程序员的可用性。

对于单元，我们已经在[程序执行](#)一章中介绍了模式 1。以下为 EM 实现 Unit3.EM4 (U3Em4_HighPerfMotionEncaps) 中的模式 1 示例。来自其他软件单元的数据通过 FB CallEquipment 的块接口传递。通过仅在最高级别连接另一个软件单元的数据源，软件单元将完全封装到另一个软件单元。来自同一软件单元内部的块的数据可直接访问。

下图中 FB CallEquipment 连接到另一个软件单元的数据。此数据通过 InOut 接口传递到 FB CallEquipment。FB CallEquipment 内部的逻辑可以使用该数据，该数据指的是接口的形式参数。

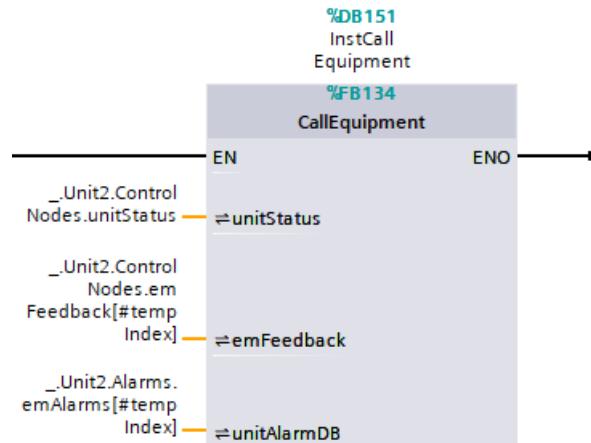


图 6-15 在软件单元的 OB Main 中调用 FB CallEquipment

下图显示了具有两个模式的 FB CallEquipment 内部的 FC。变量 emFeedback.IsLinked 和 unitStatus 通过 FB CallEquipment 的接口进行读/写。但是，DirectControl 连接到位于 EM 内部但位于 FB CallEquipment 外部的本地 DB，因此它直接访问该 DB，而无需通过块接口进行路由。它直接指向 EM 中的 DB HmiInterface，此处指向数据结构 em.directModeStateControl。这种直接的变量连接使用户可以通过工具提示“转到定义”直接跳转到程序块。

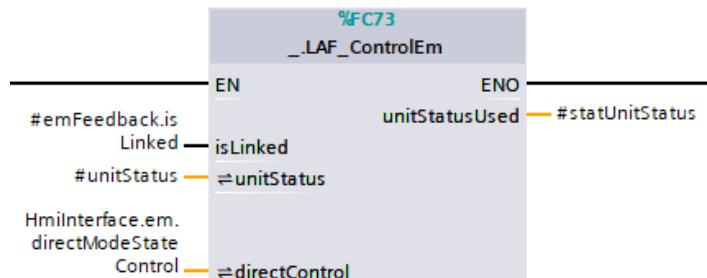


图 6-16 在 FB CallEquipment 中调用 FC LAF_ControlEm

6.7.2. PLC – HMI 握手

在 HMI 开发中，有两种方法可以处理 HMI 和 PLC 之间的按钮按下事件：

- 设置/复位模式：HMI 设置一个 PLC 命令位，PLC 处理命令，最后 PLC 复位命令位
- OnPressed 模式：只有按下 HMI 命令按钮，PLC 命令置位

两种架构各有优点和缺点：

设置/复位模式

优点	缺点
保证 PLC 接收并处理命令。	由于 PLC 必须独立于命令执行结果而复位命令，因此编程更复杂
为操作员提供更好的用户体验：无论操作员按下按钮多长时间或 PLC 需要执行命令，它最终都会被执行	增加最终导致死锁的风险，例如，如果 HMI 设置了命令，但由于任何不可预见的原因（例如中断、故障、PLC 停止/启动等）而 PLC 没有复位。同一命令的任何其他操作员命令将不起作用。

表 6-6：设置/复位模式的优点和缺点

OnPressed 模式：

优点	缺点
设置简单。	操作员可能按下了按钮，但没有任何反应，因为他按下的时间不够长，无法被下一个 PLC 更新周期检测到。
没有最终导致死锁的风险。如果操作员再次按下，将再次发送命令。	如果多个命令并行出现，则程序执行不协调的风险更高。

表 6-7 OnPressed 模式的优点和缺点

总之，设置/复位模式可实现更高的集成度和标准化。它被认为是更高级的编程技术。但是，它更复杂，必须考虑更多场景，以确保在所有情况下都能可靠运行。

在 AF 中实现

AF 可以在两种架构中进行编程。但是，AF 本身是通过 OnPressed 模式实现的，因为主要目标是提供一个通用且易于理解的框架。特别是易于设置和低编码复杂度，增加了程序员的可用性。

7. HMI 软件架构

AF 项目为 WinCC Unified 提供了一个模块化和可定制的 HMI 模板，以减少编程工作。以下章节将介绍 HMI 布局、可重用元素（如 HMI 标题）、页面导航和即插即用应用程序示例。

7.1. 画面布局

画面布局是指图形用户界面 (GUI) 画面上各种图形元素和组件的排列和设计。此布局决定了不同过程画面在 HMI 上的定位和组织方式。

画面布局在提高 HMI 的可用性和有效性方面起着至关重要的作用。

此布局（启动画面）提供了一个直观且用户友好的界面，使操作员或用户可以轻松访问相关信息和控件并与之交互。它涉及图形层级结构、对相关元素进行分组、优化空间利用率以及确保一致且美观的设计等考虑因素。

用例

在制造设施中，操作员需要一个清晰且用户友好的界面来监控各种参数、控制设备并及时响应报警。画面布局在实现这一目标方面起着至关重要的作用。

AF 画面布局代表 HMI 的起始画面，分为六个部分。

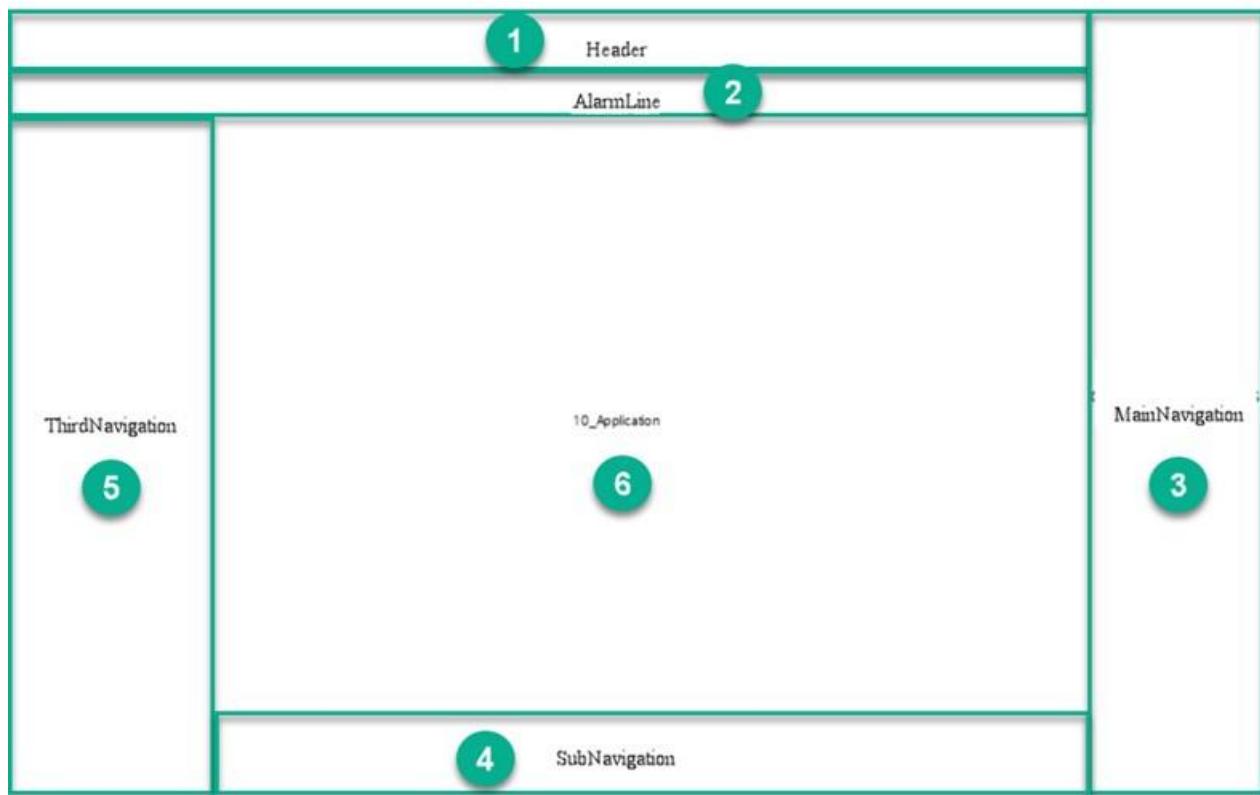


图 7-1 画面布局

部分	名称	描述
1	标题	标题始终可见，并包含最重要的信息，如 PLC 名称、当前画面名称、存储器卡信息、PN 连接、日期和时间以及 PLC 校验和。
2	AlarmLine	AlarmLine 显示最新的活动报警。
3	MainNavigation	MainNavigation 有 11 个导航区，这些区域会将用户导航到特定区域，例如：概览、状态模型、手册、消息、班次模型、生产数据、能源、CM 信息、诊断、Web、设置和用户操作。 有关“导航”主题的更多信息，请参见第 7.3 章的“导航”部分。
4	SubNavigation	SubNavigation 在画面底部提供了一个额外的导航层，可以单独定制。
5	ThirdNavigation	ThirdNavigation 在左侧提供了一个额外的导航层，可以单独定制。
6	应用	应用部分是画面的主要部分。此处包含应用程序菜单、I/O、图形和针对特定应用程序的控制变量。

表 7-1 画面布局描述

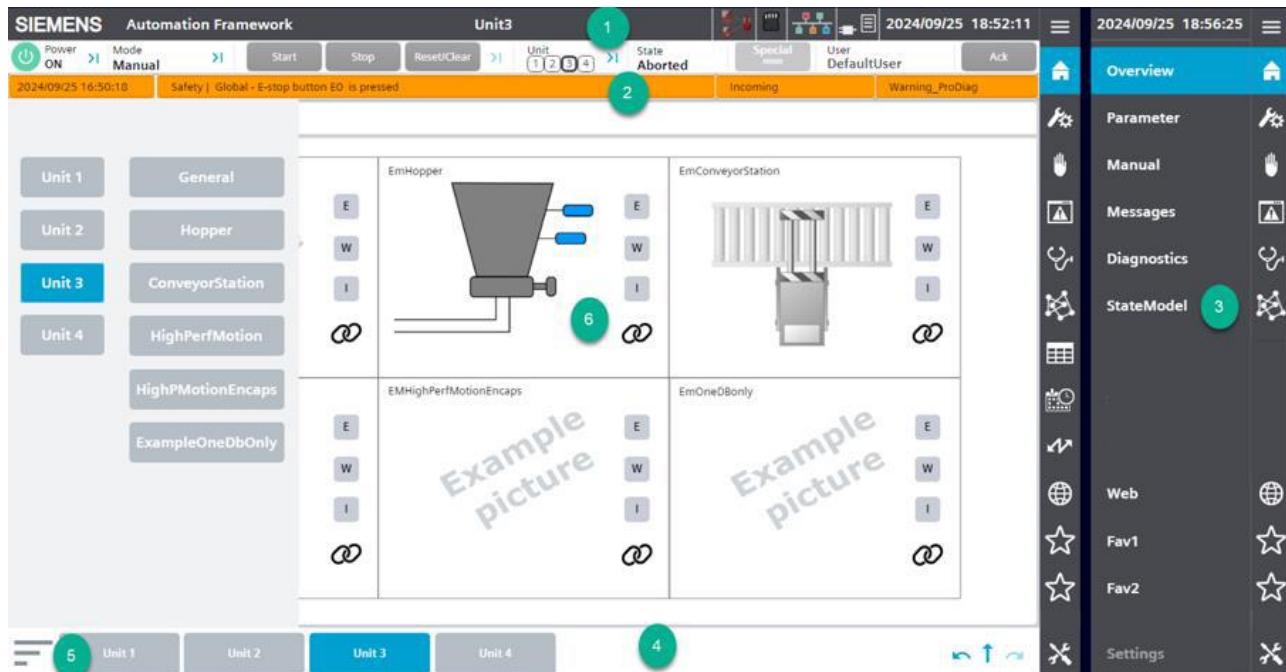


图 7-2 AF 的画面布局

第 4、5 和 6 部分是在专用的 WinCC Unified 页面中实现的，这些页面在加载启动画面时加载。其他部分在画面布局中实现，并在启动画面内进行编程。

7.2. 标题

标题显示最重要的信息：根据所选单元显示电源状态、当前状态和当前模式等。它还允许操作员控制选定的单元。

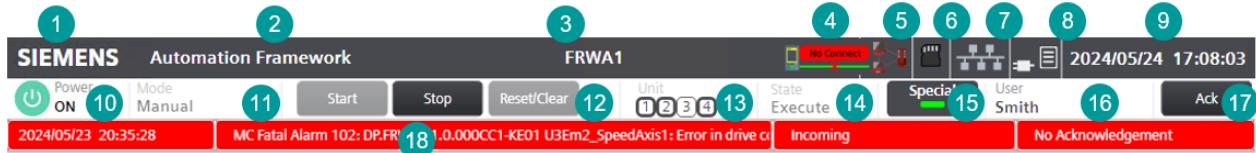


图 7-3: HMI 布局

以下图形元素位于“标题”中

元素	名称	描述
1	公司名称	公司名称的占位符。
2	PLC 名称	PLC 名称与 PLC 变量相关，并显示 PLC 的标识和维护/工厂标识数据的内容。
3	当前画面名称	使用名为“显示标题”的脚本显示当前画面的名称。
4	HMI-PLC 连接状态/ PLC 操作模式	它在 AF 项目中用于可视化 PLC 和连接状态： <ul style="list-style-type: none">PLC 停止PLC 与 HMI 之间无连接 有关详细信息，请参见 通信 一章。
6	SIMATIC 存储卡状态	有关所用 SIMATIC 存储卡的状态信息，例如存储空间大小和分配的存储器空间。包括存储卡已损耗的使用寿命以及预期使用寿命。
7	PROFINET 诊断	汇总所连接的 PROFINET 设备的诊断结果。
8	CRC 校验	标准 PLC 块、安全 PLC 块和 PLC 文本列表具有校验和。如果通过将先前的校验和与当前校验和进行比较而检测到差异，则动态 SVG 会通过图表说明这一点。
9	当前日期和时间	8.10 章节中更详细地介绍了时间 设置 。
10	电源开/关	A modal dialog box titled "Contactor Power Status". It displays the "Current Status" as "ON" with a green circle icon. Below it are two large buttons: a green "ON" button and a red "OFF" button. At the bottom is a grey "CLOSE" button.

图 7-4 电源开/关

元素	名称	描述
11	模式命令	显示当前单元模式。通过打开弹出画面，可以在单元的不同组态模式之间切换：生产，维护，手动。

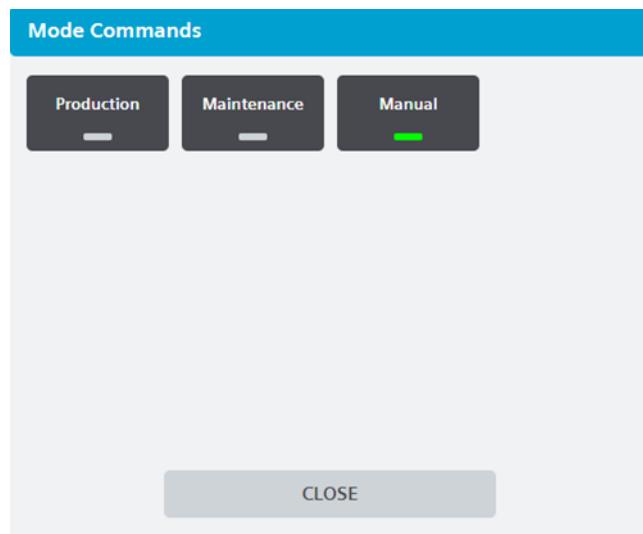


图 7-5 模式命令

12	状态命令	状态命令可以通过弹出画面传输到 PLC。
		A screenshot of a 'State Commands' dialog box. It has three rows of buttons: Row 1: Start, Stop, Reset, Clear (disabled); Row 2: Hold, Suspend, Complete, Abort; Row 3: Unhold, Unsuspend. At the bottom is a 'CLOSE' button.

图 7-6 状态命令

元素	名称	描述
13	选择单元	如果设备被分成多个单元，可以通过模式和命令选择和操作当前的活动单元。它显示每个单元的相关信息，例如模式、状态和堆叠灯状态。



图 7-7 操作单元

14	状态	显示所选单元的实际状态。当多个单元被选择并且它们处于不同状态，将显示“Various”。
15	特殊功能	操作员可在此触发对机器运行的其他要求。特殊功能可以自定义。在 WinCC 中可以通过调整文本列表来调整按钮的文本。在 PLC 的“人机界面”数据库中配置可见性和操作员控制。



图 7-8 特殊功能

元素	名称	描述
16	当前登录用户	单击用户切换登录/注销对话框，具体取决于当前用户的登录状态。

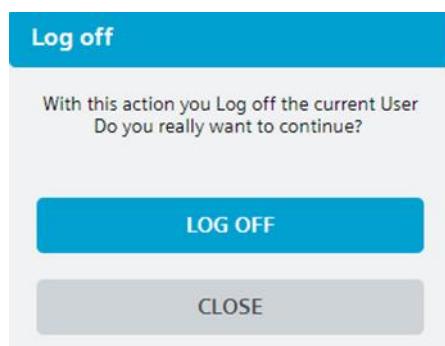


图 7-9 注销

17	确认	确认所有待处理报警的按钮。
18	AlarmLine	最后生成的报警显示在 AlarmLine 中。通过选择 AlarmLine，将显示 AlarmScreen（请参见全局脚本 AlarmLine）



图 7-10 全局脚本 – 报警

表 7-2 标题中的元素

注	在将 LAF（可能指某种特定的界面框架或库）单独用于一个新项目时，需要将“User Files → Styles Folder”中的“AF.cd19”文件复制到新项目中，以确保 HMI（人机界面）具有全部功能。		
	<table border="1"> <thead> <tr> <th>Name</th> </tr> </thead> <tbody> <tr> <td>AF.cd19</td> </tr> </tbody> </table>	Name	AF.cd19
Name			
AF.cd19			

7.3. 导航

监视参数、控制执行器、读取机器数据是工厂操作员必须在 HMI 上完成的读取/操作。因此，标准化的基本导航是高效便利操作的基础。通过导航级别创建导航，以实现统一的图形布局。

导航分为三个导航级别（图 7-1）。

- AF 画面第 3 部分：MainNavigation → 00_ScreenLayout/ScreenLayout
- AF 画面第 4 部分：SubNavigation → 00_ScreenLayout/SubNavigation
- AF 画面第 5 部分：ThirdNavigation → 00_ScreenLayout/ThirdNavigation

主导航列出了不同的要点。如果主项可以细分为多个子项，则应为此功能集成子导航。如果需要对子项进行进一步细分，则可以借助 ThirdNavigation 来实现。子导航和第三级导航的可见性可以灵活调整，而主导航永久可见。

7.3.1. 主导航

主导航由两部分组成：图标栏，永久可见，以及可以通过“汉堡”菜单打开的文本视图。两个视图都关联到 AF HMI 项目上的关键默认 HMI 功能。

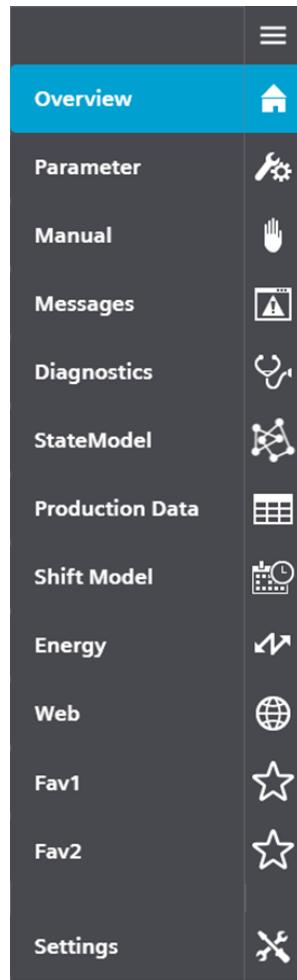


图 7-11 主导航

技术说明

这些 AF 主导航已预先组态了各种关键功能，相应地组态了其名称和图形。这些值和图形可以通过 ES 以及为每个中心功能显示的主画面（操作员每次使用主导航在画面中导航时）进行自定义。第 6 部分中使用主导航显示的画面已在 ES 上进行了

预组态。每次操作员按下这些图形和文本框以更改 swContent（画面窗口 - 第 6 部分）上的画面时，都会执行事件。如果需更改某个中心功能的画面，则必须在 ES 中修改此图形和文本框的事件。

主导航位于 ES 的 00_Screenlayout/ScreenLayout 上。导航图标位于图层 7，导航文本位于图层 6（下图）。

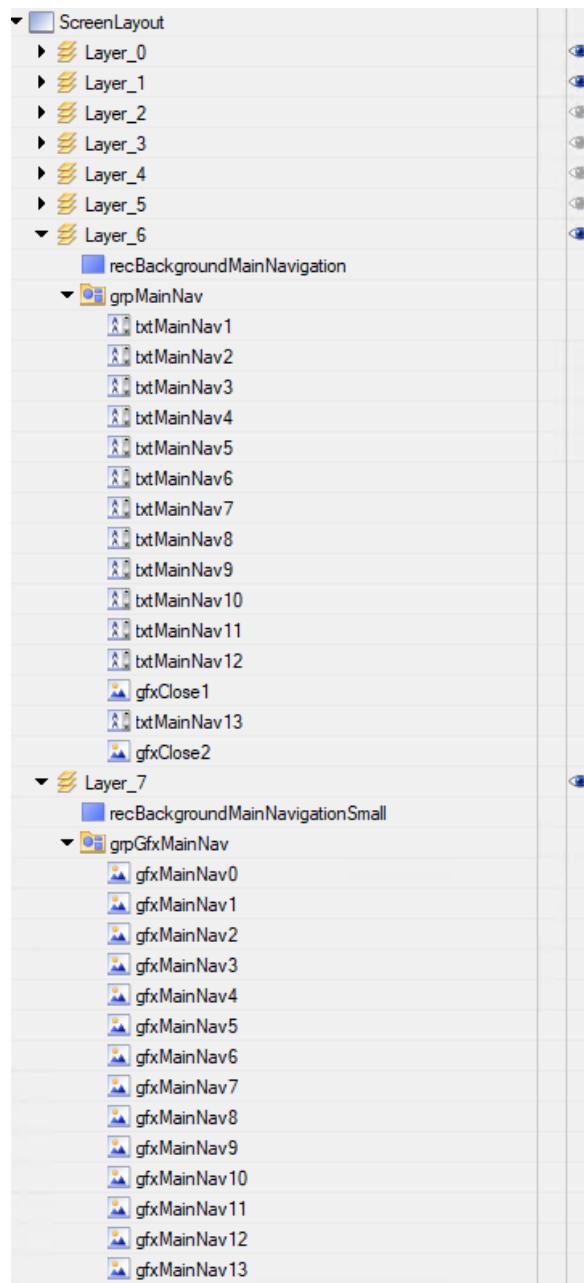


图 7-12 主导航 – Screenlayout 图层 6 和图层 7

7.3.2. 子导航

必要时，可以使用子导航来扩展主导航元素。子导航功能允许操作员在与同一中心功能（主导航）相关的多个画面中导航。为此，每个中心功能都包含一个相关的子导航。



图 7-13 子导航 – 诊断示例

技术说明

子导航涉及两种不同的场景：中心功能和单元。中心功能的子导航在 ES 上预先组态，其中其值和事件根据显示的画面进行设置，并与主导航上的每个中心功能相关。另一方面，单元和 EM 的子导航是根据项目组态动态组态的，因为单元和 EM 的数量可能因项目而异。因此，当涉及到单元和 EM 时，有一个关于子导航的特殊用例。在这些情况下，在 RT 执行期间，单元和 EM 的子导航都是动态的。

这意味着，根据 PLC 常量和 HMI 文本列表，值可以从一个单元切换到另一个单元，从一个 EM 切换到另一个 EM。因此，ES 只需要一个单元子导航和一个 EM 子导航来显示相应的值。

SubEm 导航画面由关联到每个单元背景数据块 (CallUnit_DB) 的 PLC 常量动态化，该常量连接到每个 UnitXEmNum HMI 变量（Unit1EmNum、Unit2EmNum、Unit3EmNum 和 Unit4EmNum），以根据每个单元动态显示子导航中的 EM 数量。这些 HMI 变量存储在以下 HMI 变量表中：U1、U2、U3 和 U4。这些值在每个单元的每个软件单元内的 PLC 常量上组态，如下图所示：

Unit2				
	Name	Data type	Value	Comment
▶	U2_NO_OF_EMs	Int	6	Highest Value of equipment modules in t...
▶	U2_EM0	Int	0	Equipment Module 0
▶	U2_EM1	Int	1	Equipment Module 1
▶	U2_EM2	Int	2	Equipment Module 2
▶	U2_EM3	Int	3	Equipment Module 3
▶	U2_EM4	Int	4	Equipment Module 4
▶	U2_EM5	Int	5	Equipment Module 5
▶	U2_EM6	Int	6	Equipment Module 6

图 7-14 PLC 常量 – 单元 2

它们的值（显示名称）也根据一些 HMI 文本列表（U1Em、U2Em、U3Em 和 U4Em）动态组态。

SubUnit 导航画面的值基于另一个 HMI 文本列表（单元）以类似的方式动态化。

Text lists		
...	Name	Selection
▶	U1Em	Value/Range
▶	U2Em	Value/Range
▶	U3Em	Value/Range
▶	U4Em	Value/Range
▶	Units	Value/Range
<Add new>		

...	Default	Value	Text
1.	1	General LAD	
1.	2	Template LAD	
1.	3	Template GRAPH	
1.	4	Template SCL	
1.	5	Example LAD	
1.	6	Example GRAPH	
1.	7	Example SCL	
<Add new>			

图 7-15 用于动态化 SubUnit 和 SubEMs 画面的 HMI 文本列表

7.3.3. 第三级导航

第三级导航，位于 AF 画面布局的左侧，它旨在选择要在第 6 部分（“内容画面”）中显示的特定应用内容。第三级导航中的元素遵循与子导航相同的概念。因此，它也可以动态设置。此外，在单元和 EM 之间导航时，两种导航是同步的。这意味着，操作员可以使用子导航、第三级导航或第 6 部分中显示的图形导航在画面上导航，所有这些部分都保持一致并显示相同的视图。ThirdNavigation 在某些画面情况下可以包含多个级别，例如 Home、Manual Operation、Motorstarter、Drives、IdentRFID 和 IdentMv 画面。如图 7-13 所示。其他画面仅包含一个级别。

这些级别如下：

- 常规级别或单元级别
- EM 级别
- CM 级别

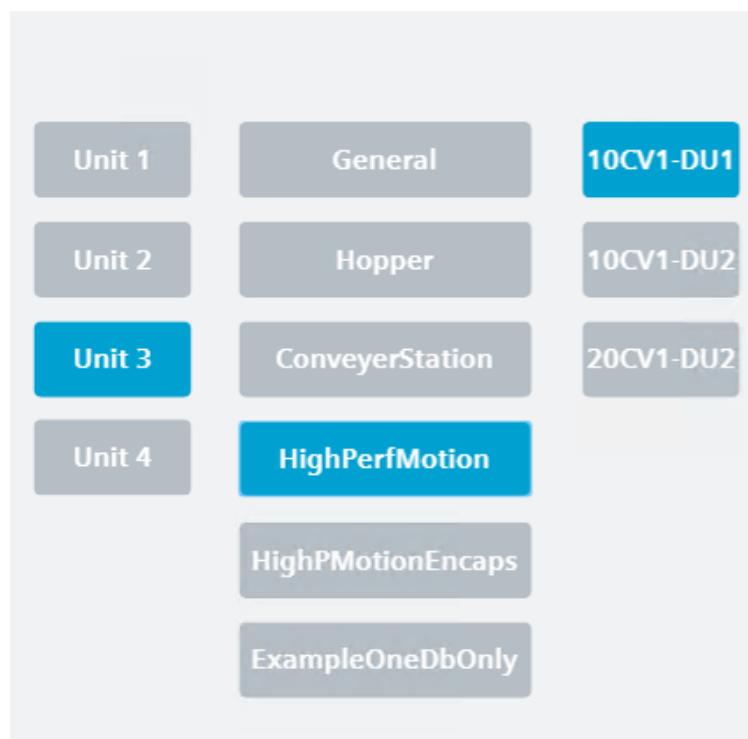


图 7-16 第三级导航

7.3.4. 单元导航

创建单元导航是为了在系统上可用的多个单元之间可视化和操作。根据用户的偏好，可以通过多种方式执行单元导航。可用选项如下：

- 应用部分： 图形导航，其中显示所有单元的概览。
- 子导航： 常规子导航，用于在单元中导航。
- 第三级导航： 常规第三级导航，用于在单元中导航。

应用部分 – 单元导航

“应用”部分可用于在单元之间导航。操作员可以单击所需的单元以显示所选单元的特定 EM。可以使用应用部分来浏览 EM 和 CM，如以下示例图片所示：

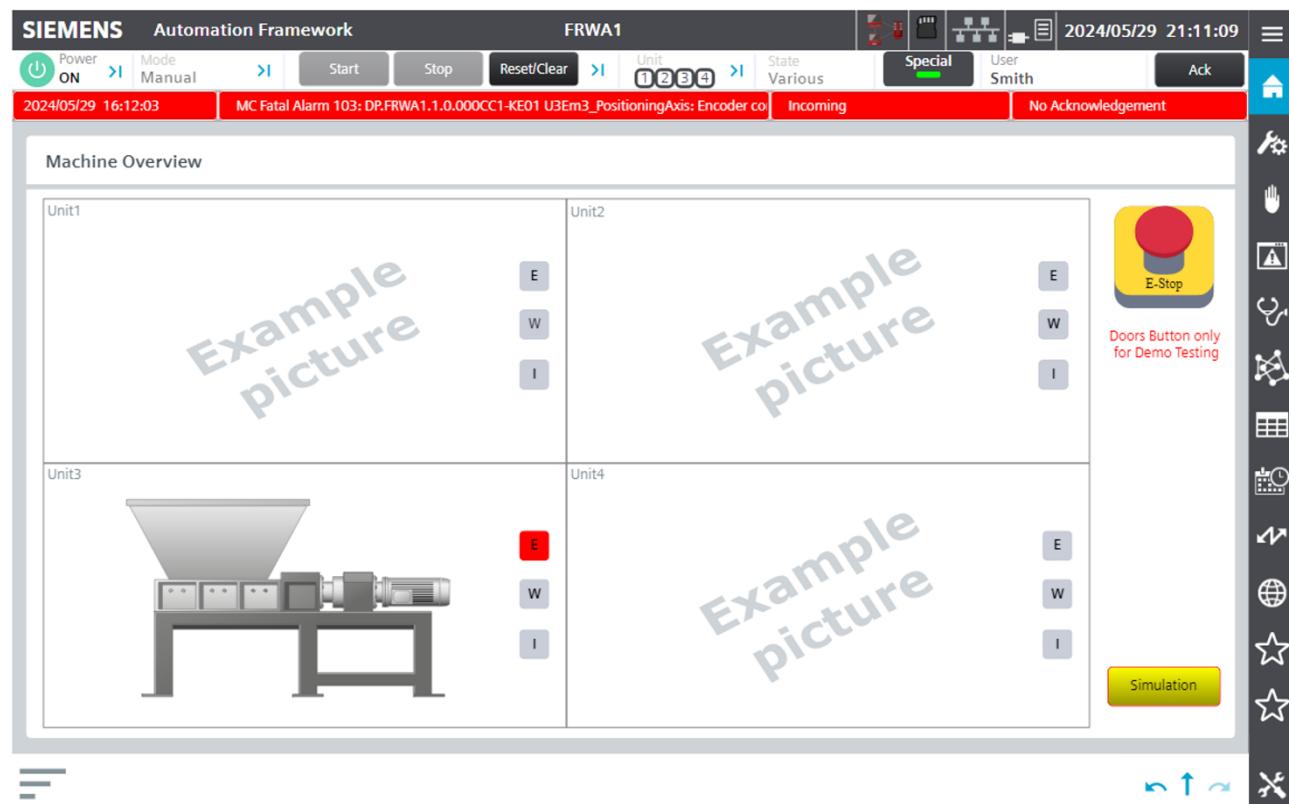


图 7-17 单元导航 – 应用部分

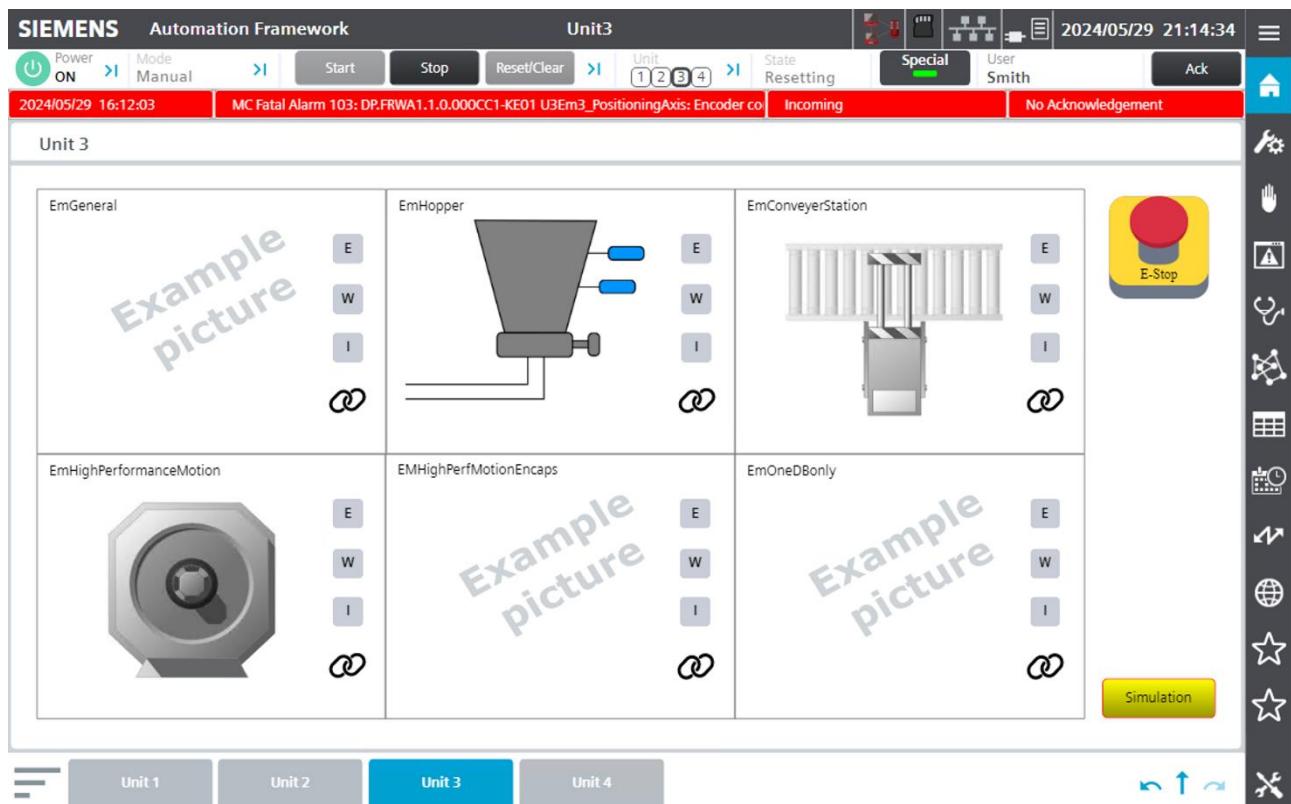


图 7-18 EM – 应用部分

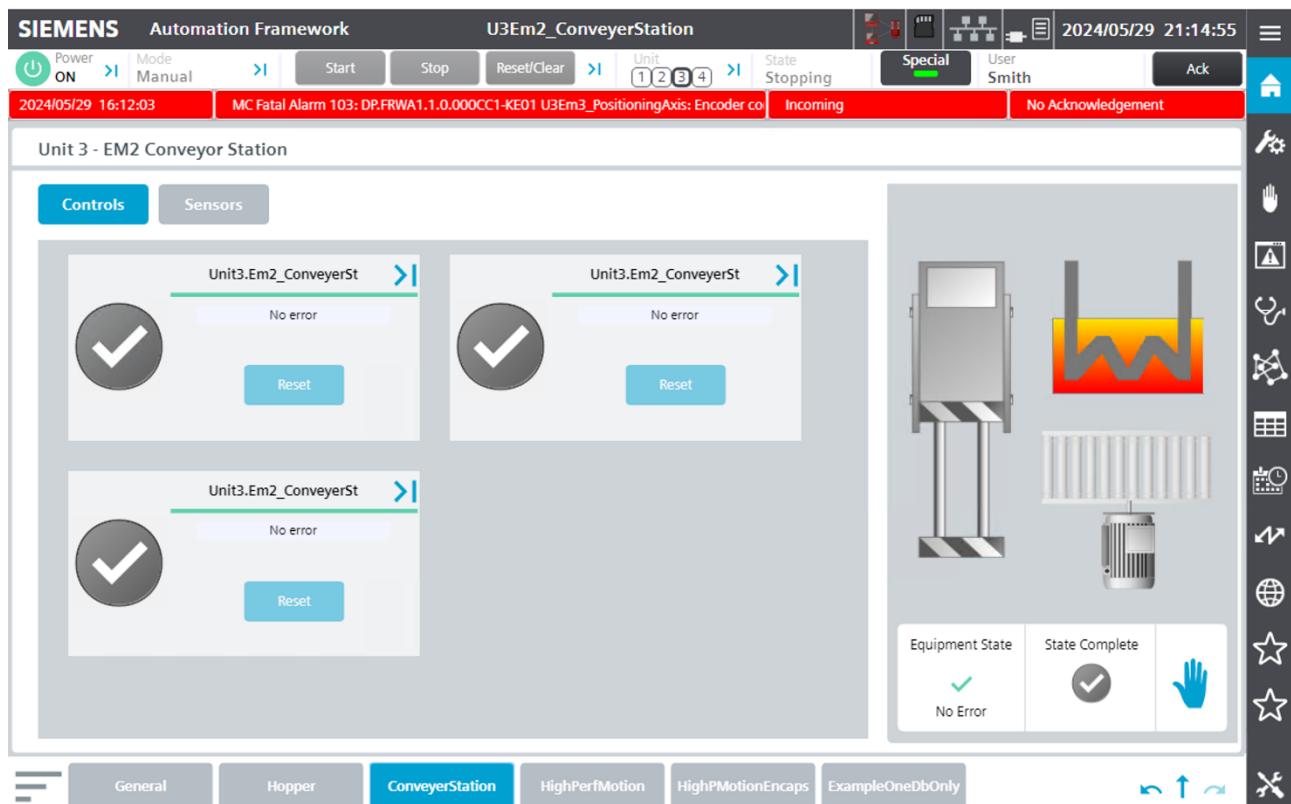


图 7-19 控制模块 – 应用部分

SubNavigation 和 ThirdNavigation – 单元导航

SubNavigation 和 ThirdNavigation 可用于以相同的方式在 EM 和 CM 中导航。操作员可以单击所需的按钮以显示该单元。通过单击多个可用按钮，可以轻松从一个单元切换到另一个单元。单元以及 EM 和 CM 导航按照相同的概念进行组态。RT 上显示的按钮名称是根据 HMI 文本列表（单元、U1Em、U2Em、U3Em 和 U4Em）和 PLC 常量动态生成的，系统在其中获取此信息并在 RT 上动态化这些值。以下图片显示了这些导航在 HMI 面板上的显示方式：

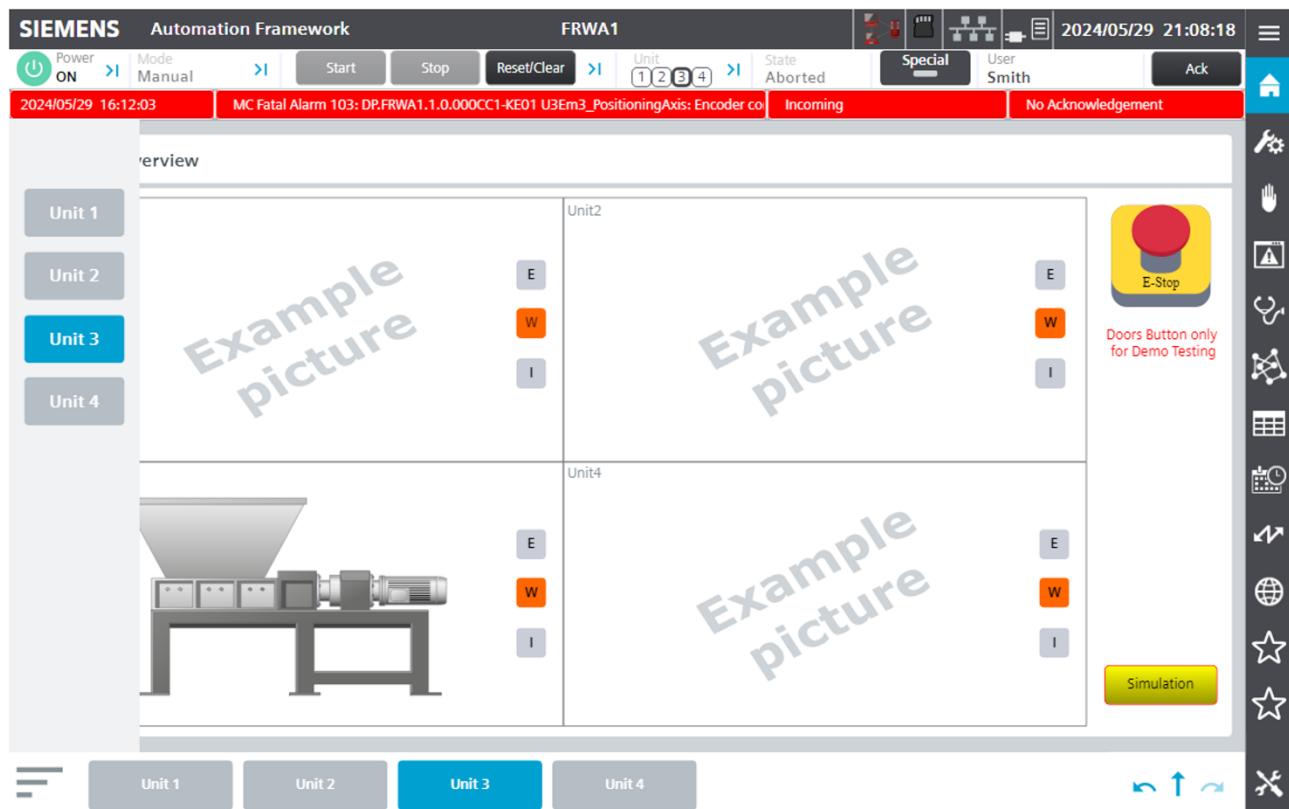


图 7-20 单元子导航和单元第三级导航

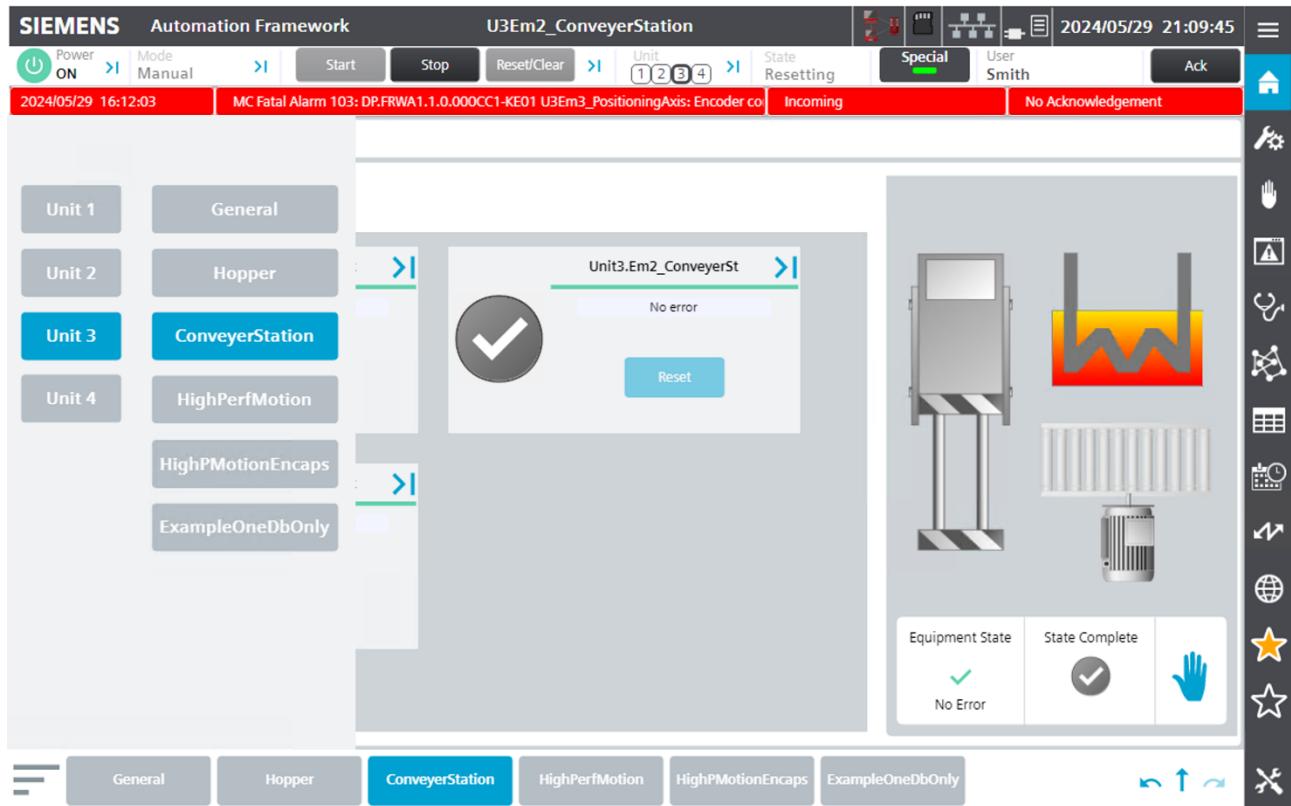


图 7-21 EM 子导航和 EM 第三级导航

7.3.5. 常规导航命令

HMI 导航包括三个额外的命令，用于在画面上导航。操作员可以使用以下命令从一个画面切换到另一个画面：上一个、向上和下一个。

- 上一个： 它将显示操作可视化的上一个画面。
- 向上： 它仅适用于单元、EM 和 CM 画面。它将显示上一级别（例如，从 CM 到 EM 或从 EM 到单元）。
- 下一个： 它将显示下一个画面。操作员应先按上一个。



图 7-22 HMI 常规导航命令 - 上一个、向上和下一个

7.3.6. HMI 项目树导航

项目树导航基于前几节中介绍的多种导航类型。画面和 HMI 变量遵循相同的结构。此结构如下图所示：

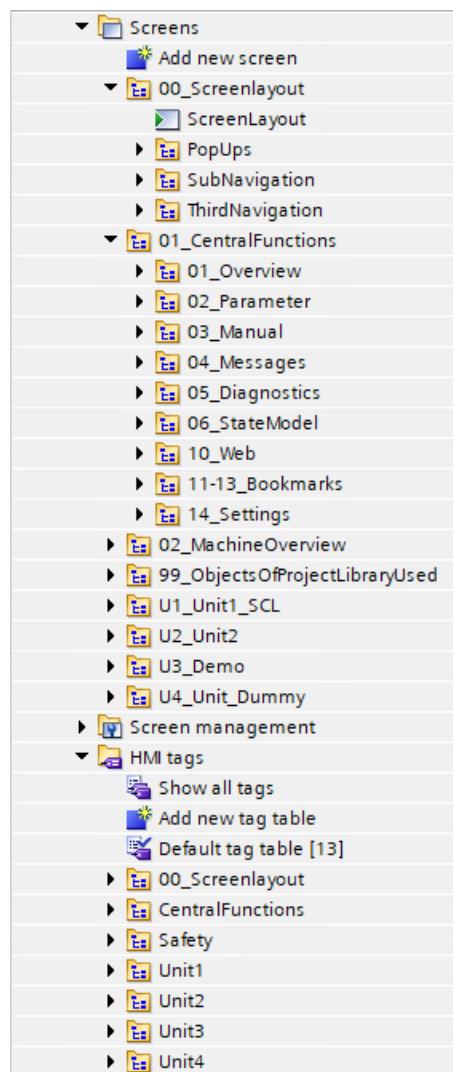


图 7-23 项目树结构 – 画面和 HMI 变量

7.4. 机器概述

机器概览屏幕是人机界面上机器可视化的最高级别。由于 AF 项目支持多个单元，因此最高级别的屏幕就是下面的概览屏幕。此外，还添加了其他元素，如每个单元的当前活动错误 (e)、警告 (w) 和信息 (i) 指示器等。

ISA-88 机器结构以屏幕的树形结构表示。点击一个单元，屏幕就会跳转到下一级，显示该单元的所有设备模块。

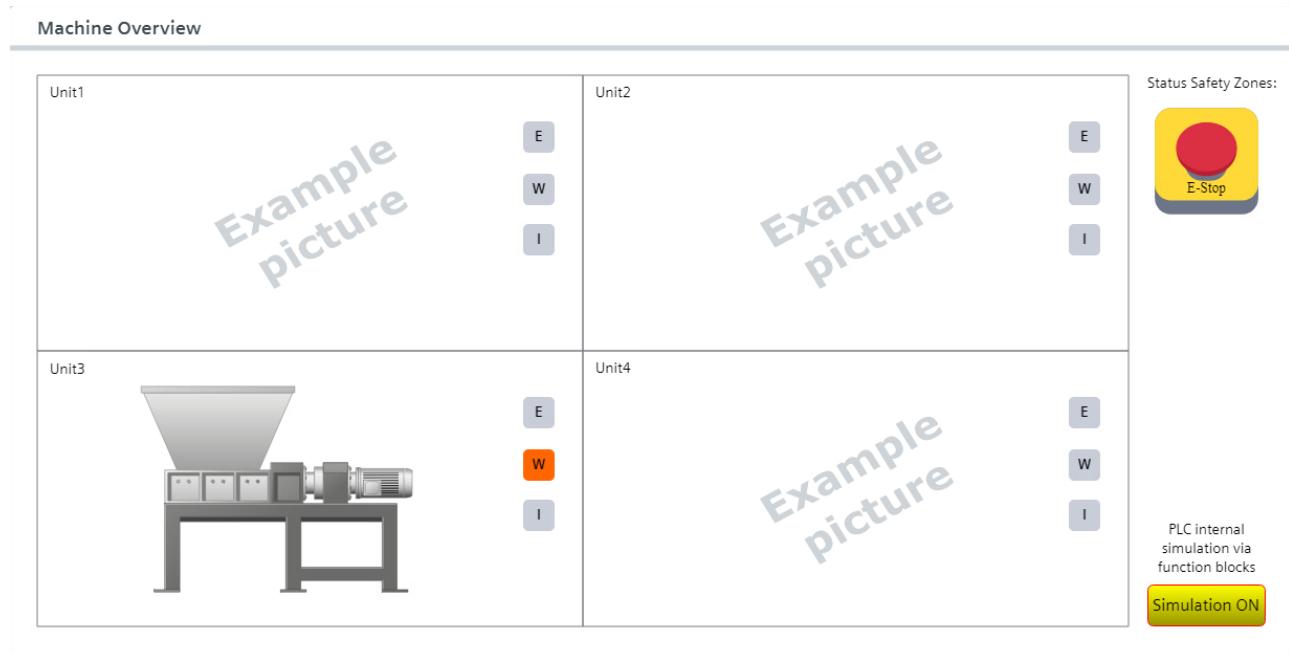


图 7-24 机器概述样例

7.5. 操作锁定机制

如果通过多个 HMI 来控制机器，只允许同时从其中一个 HMI 控制机器。这可以通过使用操作锁定功能来实现。

概念

基本上，所有 HMI 都将从锁屏开始。

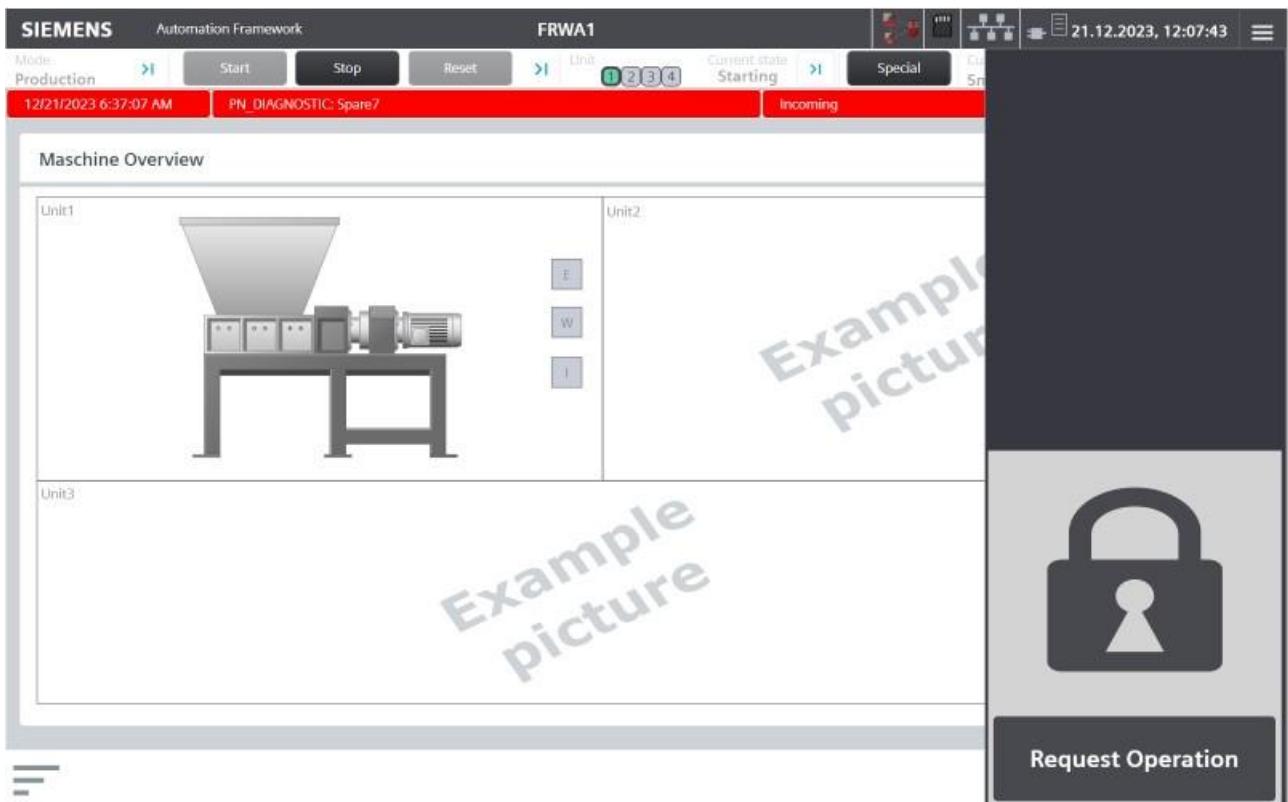


图 7-25 操作锁定

当机器通电时，连接到本机的所有 HMI 设备将显示常规操作锁定侧边栏窗口。操作员需要按下“请求操作”按钮来控制机器。HMI 上的操作锁定指示“无”或“另一个”站正在操作机器，并且可以请求机器的常规操作权限。

如果操作员在另一台 HMI 上按下“请求操作”按钮，主控操作员将立即被中断并失去控制机器的权利。此默认行为可以自定义。

技术说明

在 HMI 画面“00_ScreenLayout”中，优先级最高(31)的是覆盖整个画面区域的“swLock”覆盖画面。当按下“请求操作”按钮时，本地 HMI 的以下变量将：

- 从 @ServerMachineName 复制到 00_Screenlayout\Hmi_operation\Interlock\commands\ “ServerMachineName”
- 从 @LocalMachineName 复制到 00_Screenlayout \Hmi_operation\Interlock\commands\ “LocalMachineName”

swLock 画面的可见性由脚本控制，该脚本由连接到 HMI 变量的 PLC 变量

(Hmi.operatorInterlock.commands.serverMachineName 和 Hmi.operatorInterlock.commands.localMachineName) 触发，这些 HMI 标签基于 HMI 面板的 @ServerMachineName 和 @LocalMachineName。这意味着当操作员按下请求操作按钮时，HMI 会使用其值更改这些变量，并授予对此特定 HMI 设备的访问权限。此时，其他 HMI 设备被锁定（因为 ServerMachineName 和 LocalMachineName 不再与 PLC 中心值匹配）。

在工程组态或调试阶段，可以禁用 swLock 功能。为此，必须用 WString “OFF” 覆盖 PLC 变量“Hmi.operatorInterlock.commands.serverMachineName”

注

建议仅通过 TIA Portal 在线手动执行此操作。请勿更改起始值。

■ ▼ operationInterlock	AF_typeOperationI...	Module interface for external systems
■ ▼ commands	AF_typeOperationI...	Module related commands from external systems
■ reqMaintenanceLock	Bool	TRUE: Request maintenance lock from HMI
■ reqMaintenanceReset	Bool	TRUE: Request reset of maintenance lock from HMI
■ serverMachineName	WString	@ServerMachineName tag of the panel currently in Operation
■ localMachineName	WString	

图 7-26 软件单元 CentralFunctions 中 DB Hmi 中的 OperationInterlock 数据结构

8. HMI 中央功能

中央功能存在于 PLC 和 HMI 中。在 PLC 中用软件单元表示，在 HMI 中用文件夹表示。全局 "中的文件夹结构和文件夹编号与 PLC 和 HMI 一致，便于查找相关文件夹。本章将对 PLC 部分和 HMI 部分进行说明。

中央功能包含影响整个系统的全系统功能，或高于单元级别的分级功能。

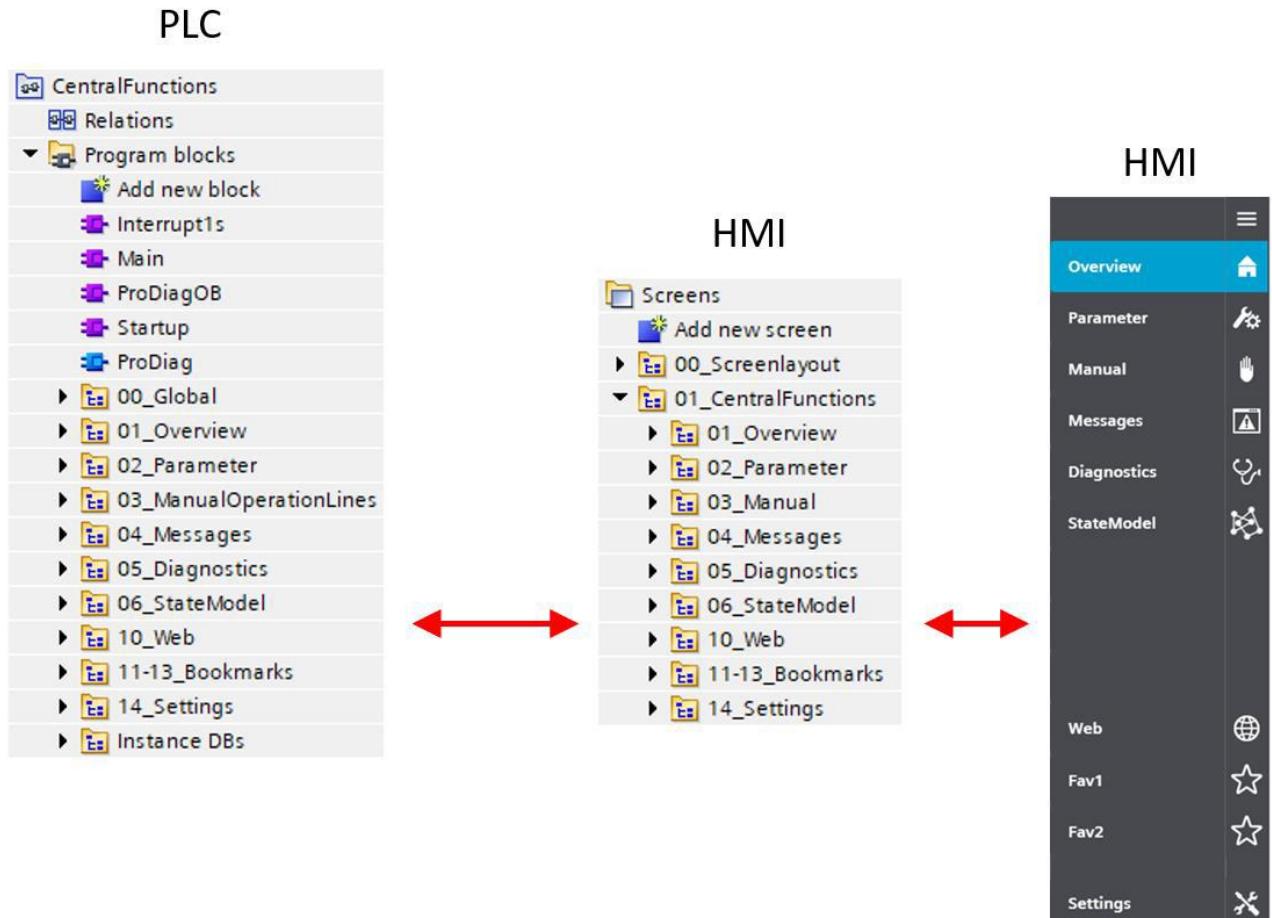


图 8-1: PLC 和 HMI 中的中央功能区

8.1. 全局

在 PLC 中，“全局”包含系统功能、人机界面处理和模拟数据。具体包括：

- 全局确认
- 全局电源开/关
- 与人机界面主接头的连接
- 本地 PLC 日期和时间
- 时钟位生成
- 与 NTP 进行时间同步



图 8-2: PLC 中对应的内容

在 HMI 中, "Central Functions" 下没有 "Global" 文件夹, 因为 PLC 功能与屏幕 "ScreenLayout" 中的 HMI 主标题相关。



图 8-3: AF HMI 标题

人机界面页眉显示最全面的信息: 根据所选单元显示电源状态、当前模式和状态。操作员还可以通过它发送命令, 切换电源状态、模式和发送命令。此外, 还可以选择特殊功能以及全局确认命令。[命令和数据流](#)一章介绍了命令集成。

主标头与 DB “Hmi” 相连。FB “HmiHeaderManager” 负责处理人机界面和设备之间的数据流。它可以处理多个面板和多个单元。根据在人机界面中选择的单元, 操作员通过按钮下达的命令会分配到相应的单元。反之亦然, 根据在人机界面中选择的单元, 按钮外观和数值显示(模式、状态)将从单元汇总到 PLC 并发送到人机界面。FB “HmiHeaderManager” 负责处理所有这些工作。它可以协调多个单元的多个人机界面。

FC “CallSimulation” 可自动检测是否连接了 SIMIT 仿真, 并相应设置不同的位。

8.2. 概览

人机界面中的概览文件夹包含用于机器概览屏幕的弹出窗口。AF 提供两个弹出窗口:

- InterlinkControl (互连控制): 使用用户能够从设备上链接或取消链接设备模块。
- InterlinkControl_NonUnlinkable: 设备模块无法解除链接的信息框。

在 PLC 中, 与这些弹出窗口相关的功能由设备模块的 FB “CallEquipment” (调用设备) 处理。

8.3. 参数 (主导航)

在参数画面中, 可以实现各种组态用例, 例如调试设置、配方等。在 AF 项目中, 画面使用 WinCC Unified 的“参数集控制”对象。

参数控制是方便组态工程师、操作人员和配方创建者控制参数集的综合功能。参数控制具有以下优势:

- 可以为一个或多个参数集类型应用用户数据类型的结构。
- 可以通过新用户数据类型版本自动更改一个或多个参数集类型的结构。
- 可以手动或自动在 HMI 设备和 PLC 之间交换大量参数, 以部署生产机器。
- 可以在工程组态期间或持续操作期间轻松为要生产的产品统一创建参数集。
- 通过构建相关的参数/设定值, 可以轻松传输参数。

Parameter set type Number

No elements available

Parameter set Number

No elements available

	Name	Value	Unit of measurement
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			

Parameter set type is not available.

图 8-4 参数集控制



如需更多详细信息，请参阅 WinCC Unified 文档中的“创建和配置参数集控制”部分：

<https://support.industry.siemens.com/cs/ww/en/view/109821886>

8.4. 手动

手动画面可以实现手动操作 CM 的各种用例。为此，AF 引入了手动操作行，这是一种简单的解决方案，具有极强的定制能力，可以通过两个按钮控制 CM，并显示有关终端开关和/或当前速度和位置等值的信息。下图显示了手动操作行的示例：

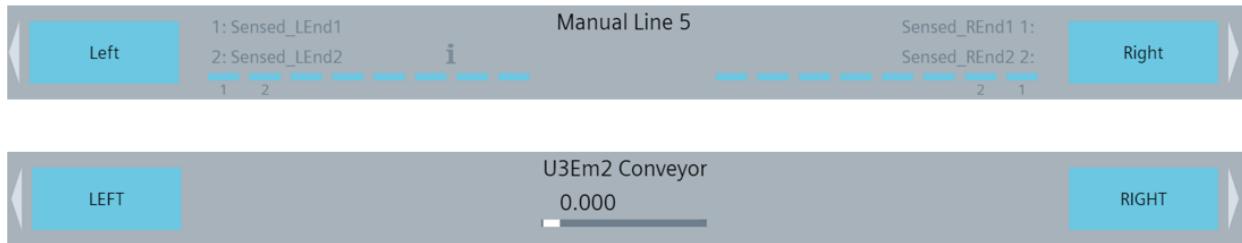


图 8-5 手动操作行示例

使用手动操作行主要有两个用例：

- 使用现成的通用画面“LAF_ManualOperation1_1280x800”创建 CM 的概览，这在添加新 CM 时更易于使用，因为大多数 HMI 都可以在 AF 中直接使用。
- 在自定义画面中使用面板，这需要更多的准备工作，因为需要手动将准备好的功能添加到画面上才能正常运行。

8.4.1. 手动操作行入门

本章介绍手动操作行所需的组态元素。详细的分步示例可以在下一章[框架演示实现](#)中找到。

适用于所有手动操作行的全局 DB

内部使用手动操作行时，必须创建一个名为“LAF_InternalManual”的全局 DB 和“LAF_typeManHMIDisp”数组 [1..1]。不需要组态。

设备模块 DB

按照 AF 编程建议，每个 EM 都应该有一个专用的 DB，用于使用手动操作行操作的每个 CM。可以在“03_Manual”文件夹下的“中央功能”软件单元中找到相应的模板 DB “ManualOperationControl_Template”。数据结构由三个元素组成：

- 类型为“LAF_typeManEm”的 EM 组态“manualOperationEm”
- 类型为“Array[1..<X>] of LAF_typeManOperationArea”的数组“manualOperationEmArea”，其中 X 表示对手动操作行进行分组的区域数
- 类型为“Array[1..<Y>] of LAF_typeManOperation”的数组“manualOperationCms”，其中 Y 是 EM 中的 CM 数

使用“manualOperationEm”时，只需组态“equipmentModuleNum”组态。“equipmentModuleNum”具有用于识别 EM 的值。由于 EM 是单元的一部分，因此建议使用右侧的第二位和第三位数字表示 EM 编号，使用右侧第四位和第五位数字表示单元编号。例如，区域 5 单元 3 中的 EM 2 值为“3025”。由于数据类型为<UInt>，因此最大值不应超过 64999（单元：64，EM：99，区域：9）

对于“manualOperationEmArea”，数组中每个区域的唯一设置是<string>类型的“displayedLines”。可以组态字符串，以便可以在该区域中组态手动操作行的各种组合。例如，要显示第 1 个行到第 6 个行，请使用格式“1-6;”。使用逗号进行排序，例如“6,5,1,8”以 6->5->1->8 的顺序显示 CM。如果一个区域有 6 个以上的 CM，则

“LAF_ManualOperation1_1280x800”画面中将有一个上下页面。

对于“manualOperationCms”，唯一的组态是通过设置数组大小来计算 CM 的数量。

注

与不同软件单元中的所有组态设置类似，建议在启动 OB 期间设置上述手动操作行组态。

文本列表

文本列表用于两种不同的功能：

- 手动行的颜色组态
- EM 的区域名称

颜色的文本列表的格式为“LAF_manualColor<X>”，其中 X 是 Json 文件中所选内容的编号，例如“LAF_manualColor1”。颜色文本列表具有固定格式，值为 1 – 正常状态、2 - 按下、3 - 确认、4 – 禁用和 5 - 停用 + 按下，颜色值在“文本”中使用十六进制码进行设置。十六进制码中“0x”后的前两位数字是透明度设置，“ff”表示 0% 透明，“00”表示 100%，最后 6 位数字是正常颜色十六进制码。

区域名称的文本列表的格式为“EquipmentAreas_<X>”，其中 X 是设备区域代码，例如“EquipmentAreas_3020”。此文本列表用于画面“LAF_ManualOperation1_1280x800”，下图中显示的区域具有相应的文本，具体取决于 EM：



图 8-6 每个 EM 的区域名称

Json 文件

每个控制模块/手动操作行都需要一个 Json 文件。每个 Json 文件都包含有关名称、值单位、结束开关数量、按钮颜色、结束开关颜色和每个结束开关名称的信息。Json 文件名的格式为“<X>_<Y>_<Z>.json”，其中 X 为 EM 编号，Y 为行号，Z 为语言代码。Json 文件名的一个示例为“3020_1_1033”，第 1 个行的 EM 编号为 3020，语言为英语（由区域设置 ID 标准定义为 1033）。

需要手动将 Json 文件添加到 PC Runtime 或面板中，放置路径在 HMI 变量中组态为“manualOperationFilePath”。

注

使用 Windows PC Runtime 路径字符串时，必须包含其他转义字符，如下所示：“D:\\JsonFiles\\”（双反斜杠）。

使用 Unified 面板时，路径应为“/home/industrial/JsonFiles/”，不带额外的转义字符。

Json 文件需要通过 U 盘传输到 Unified 面板。在面板中，可以打开应用程序“文件浏览器”，并且可以将 Json 文件夹复制并粘贴到相应的路径。要在“文件浏览器”中复制项目，请使用菜单“编辑”。

FC 调用

使用手动操作行时，有两个 FC 是强制性的：

- “LAF_manEmConfig” 和
- “LAF_manOperationCm”。

必须使用以下组态：

- LAF_manEmConfig

本章开头介绍的 EM 组态 DB 的三个分支（“manualOperationEm”、“manualOperationEmArea”、“manualOperationCms”）必须连接到 FC “LAF_manEmConfig”。此外，用于画面数据的 DB “LAF_InternalManual”必须连接到 FC 才能管理“LAF_ManualOperation1_1280x800”画面。下图显示了此类组态的示例。

Network 4: Unit 3 Equipment Module 2 Configuration

Equipment Module 3020

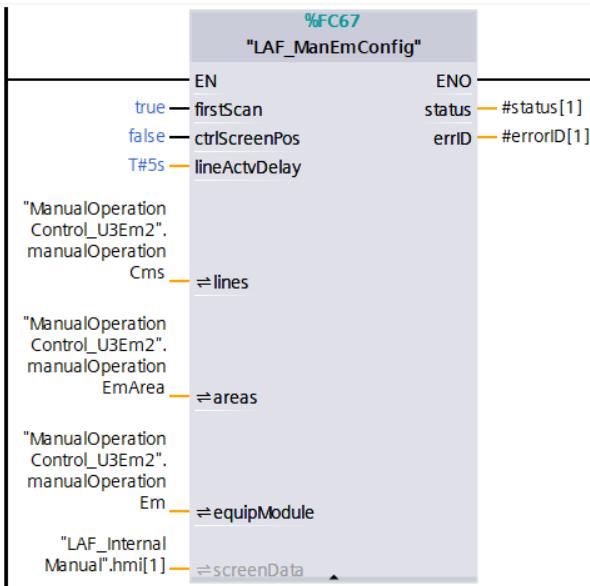


图 8-7 为 Unit3 的 EM 2 组态的块“LAF_manEmConfig”

- LAF_manOperationCm

使用“LAF_manOperationCm”，连接用于 CM 控件的变量。每个 CM 只能连接到一个“LAF_manOperationCm” FC。块文档中介绍了所有 FC 输入，以下示例中仅介绍一些基本功能：

输入	描述
“rightEndPosSens<X>”和 “leftEndPosSen<Y>”	来自过程的关于终端开关处于打开或关闭状态的反馈
“execRightCmd”和“execLeftCmd”	像物理按钮一样控制 CM 的硬件
“cnfrmExecRightCmd”和 “cnfrmExecLeftCmd”	正在执行命令的反馈
“expandable”	打开特定控制节点的画面窗口。将要打开的画面需要使用名称 “ControlScreen_<X>_<Y>”，其中 X 是 EM 编号，Y 是行号。
...	

表 8-1 选择“LAF_manOperationCm”输入

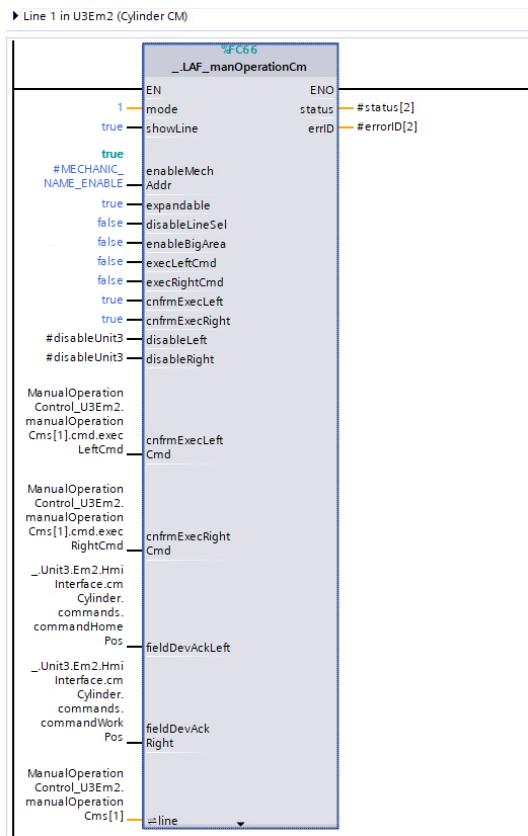


图 8-8 Unit3 的 EM2 中手动操作行 1 的“LAF_manOperationCm”组态示例

8.4.2. 框架演示实现

在 AF 项目中实现的手动操作行位于主导航中的手动操作下。

在 AF 项目中，所有用于手动操作行的库块和 UDT 都已添加到位于文件夹“LAF - 90_ManualOperationLine”内的软件单元“ObjectsOfProjectLibraryUsed”中。所有 UDT 和块都已发布，以便可以从需要它们的软件单元访问它们。由于它们是主导航栏的一部分，因此在 AF 项目中，手动操作行已在“CentralFunctions”软件单元中组态，位于名为“03_Manual”的文件夹中，通过该文件夹调用这些手动操作行。

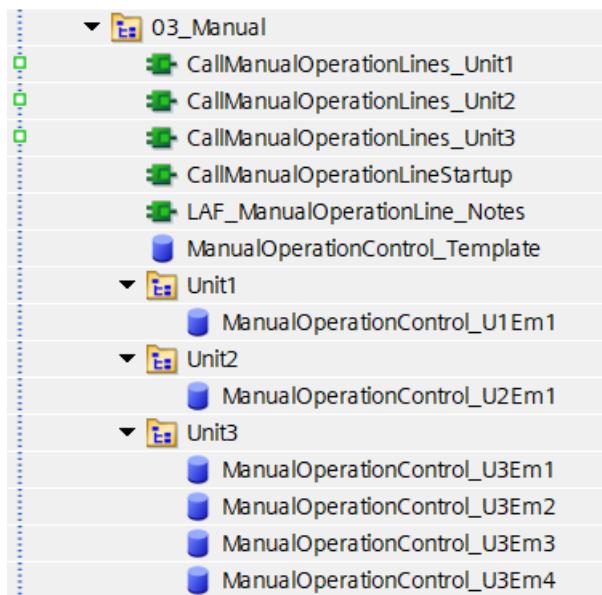


图 8-9 为手动操作行编程的块

基本组态

创建“CallManualOperationLines_UnitX”FC是为了调用以下块：每个EM的“LAF_manEmConfig”和每个行的“LAF_manOperationCm”块，需要按照上文[手动操作行入门](#)的说明进行组态。

“CallManualOperationLines_UnitX”中的FC的连接是通过DB完成的。每个EM都有一个用于组态、区域组态和CM的DB。每个ManualOperationControl DB都具有单元号和EM号作为扩展名，并且位于具有它们所属的单元名称的文件夹中。它们都是基于模板DB“ManualOperationControl_Template”创建的。下图显示了如何组态Unit3的EM2的一个DB。有别于模板的更改内容以红色突出显示。包括：

- 具有正确命名法的EM编号(3020)。
- 每个区域要显示的行数，此处为第一个区域的六行。

ManualOperationControl_U3Em2			
	Name	Data type	Start value
1	Static		
2	manualOperationEm	"LAF_typeManEm"	
3	equipmentModule...	UInt	3020
4	actvArea	UInt	0
5	pages	"LAF_typeManPageDetails"	
6	manualOperationEm...	Array[1..7] of "LAF_typeManOperationArea"	
7	manualOperation...	"LAF_typeManOperationArea"	
8	areaNum	UInt	0
9	displayedLines	String	'1-6.'
10	hmiPage	"LAF_typeManPageDetails"	
11	manualOperation...	"LAF_typeManOperationArea"	
12	manualOperation...	"LAF_typeManOperationArea"	
13	manualOperation...	"LAF_typeManOperationArea"	
14	manualOperation...	"LAF_typeManOperationArea"	
15	manualOperation...	"LAF_typeManOperationArea"	
16	manualOperation...	"LAF_typeManOperationArea"	
17	manualOperationCms	Array[1..10] of "LAF_typeManOperation"	

图 8-10 手动操作 DB 组态示例 U3 EM2

“CentralFunctions”软件单元中的“CallManualOperationLineStartup”FC用于设置EM的手动操作行组态，确保启动后组态不会为空。

连接 CM

组态DB后，将连接添加到“CallManualOperationLines”内部的块中，将多个输入连接到EM的相应控制节点DB。例如，对于Unit3中EM2的Line1，连接如下：

- 输入“cnfrmExecRightCmd”和“cnfrmExecLeftCmd”用于获取反馈，即“按钮正在按下”，PLC获得此信息。简单的实现是从接口中使用“execRightCmd”和“execLeftCmd”，类似于“ManualOperationControl_U3Em2.manualOperationCms[2].cmd.execLeftCmd”。
- 输入“leftEndPosSens1”连接到“_.Unit3.Em2.HmiInterface.cmCylinder.monitoring.inHomePos”，这是参数“inHomePos”的位置，该参数来自Unit3中EM2的气缸示例，该气缸正在通过此特定的手动操作行进行控制。
- 输入“fieldDevAckLeft”和“fieldDevAckRight”是现场设备执行的确认，它连接到“_.Unit3.Em2.HmiInterface.cmCylinder.commands.commandHomePos”，当激活了转到主位置的命令时，它来自气缸的控制节点。

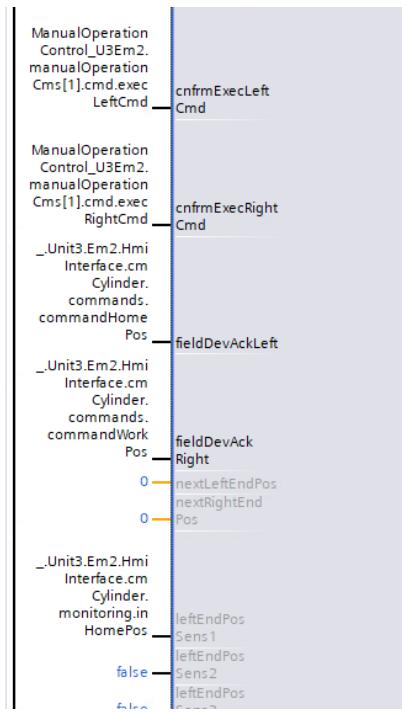


图 8-11 仔细查看“LAF_manOperationCm”中组态的输入

一旦在“CallManualOperationLines”中设置了 EM 的所有组态块和每个行的块，PLC 中手动操作行的实现就完成了。

HMI 实现

首先，通过名为“03_manualOperation”的预定义表组态 HMI 变量，与相应的 PLC 变量建立连接，该变量将来自位于软件单元“CentralFunctions”内文件夹中的 DB “LAF_InternalManual”，该变量将显示来自 HMI 接口的数据。

如第 9.5.1 章所述。在 HMI 变量表中，需要指定 json 文件的路径。需要修改的变量为“manualOperationFilePath”，路径应在 Properties>Values 中指定（见图 8-12）。由于该项目使用 Unified 面板，因此路径值为“/home/industrial/JsonFiles/”，如下图所示，初始路径设置为“D:\JsonFiles\”，以便在仿真中进行测试。当下载到面板时，“ScreenLayout”画面的已下载事件会自动将 json 文件路径更改为“/home/industrial/JsonFiles/”。

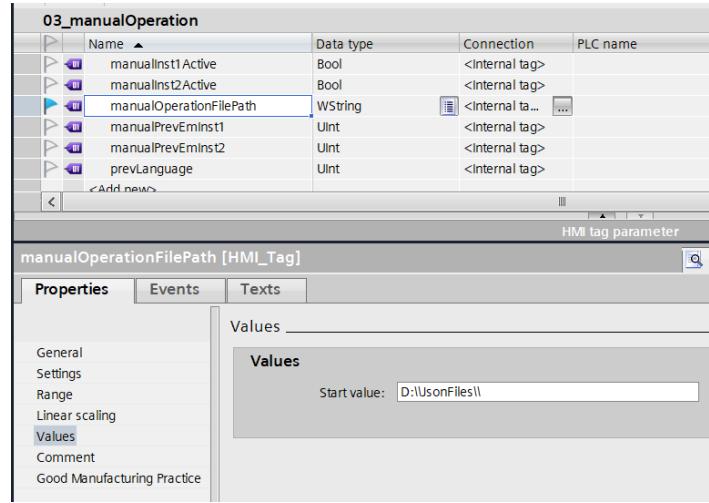


图 8-12 HMI 变量“manualOperationFilePath”组态

接下来添加文本列表。在 AF 项目中，将有两种不同类型的文本列表会影响手动操作行的使用：

- “EquipmentAreas”文本列表：需要为每个 EM 组态具有相应名称的 EM。例如，在“EquipmentAreas_1010”文本列表中，区域的组态如下图所示：

Value	Text
1011	Area 1
1012	Area 2
1013	Area 3
1014	Area 4
1015	Area 5
1016	Area 6
1017	Area 7
1018	Area 8
1019	Area 9
1020	Area 10

图 8-13 “EquipmentAreas_1010”文本列表

- “LAF_manualColor” 文本列表：它们是 0 到 16 的文本列表，其中以十六进制值的形式包含手动操作行面板中使用的颜色。需要这些文本列表来转换从 json 文件读取的颜色。

Value/Range	Areas
Value/Range	EquipmentAreas_1010
Value/Range	EquipmentAreas_2010
Value/Range	EquipmentAreas_3010
Value/Range	EquipmentAreas_3020
Value/Range	EquipmentAreas_3040
Value/Range	EquipmentAreas_3030
Value/Range	IdentAlarmCategory
Value/Range	IdentAlarmMv
Value/Range	IdentalarmRf
Value/Range	LAF_GMReasons
Value/Range	LAF_manualColor0
Value/Range	LAF_manualColor1
Value/Range	LAF_manualColor10
Value/Range	LAF_manualColor11
Value/Range	LAF_manualColor12
Value/Range	LAF_manualColor13
Value/Range	LAF_manualColor14
Value/Range	LAF_manualColor15
Value/Range	LAF_manualColor16
Value/Range	LAF_manualColor2
Value/Range	LAF_manualColor3
Value/Range	LAF_manualColor4
Value/Range	LAF_manualColor5
Value/Range	LAF_manualColor6
Value/Range	LAF_manualColor7
Value/Range	LAF_manualColor8
Value/Range	LAF_manualColor9
MC V8.0	LAF_tIDriveDiagAxis_AxisSensorAdaption

图 8-14 框架演示中所需的手动操作行文本列表

接下来，必须将手动操作行的脚本添加到 HMI 项目中。有两个不同的文件夹，其中包含多个必需的脚本，以便行正常工作：

- 第一个是“ManualOperation”文件夹，其中包含画面内部使用的主要脚本
- 第二个是“ManualOperationAddrMode”文件夹，其中包含在必要时显示其他寻址模式（如电气寻址和机械寻址）的脚本。

注

寻址模式：

如果在中间触摸手动操作行，最多可以显示三个不同的文本（符号、电气或机械）。

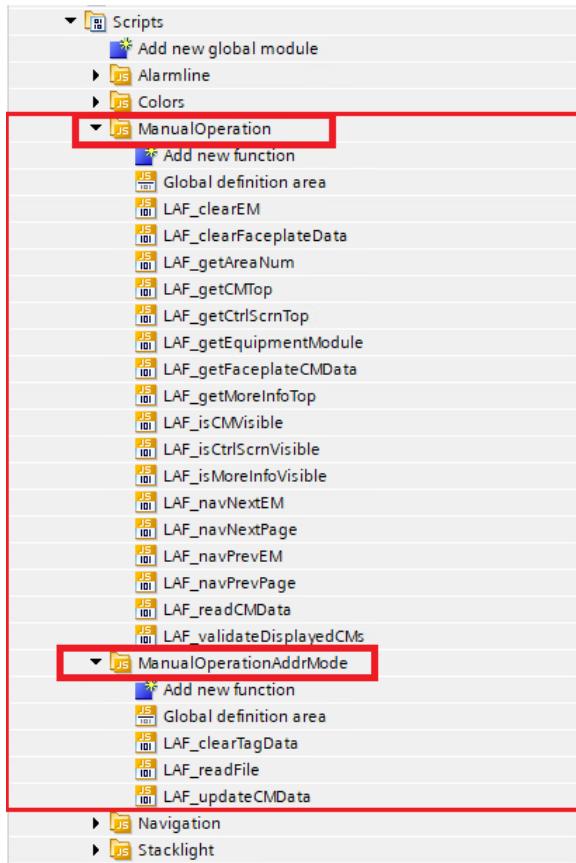


图 8-15 手动操作行脚本

接下来，为手动操作行创建画面。在本例中，它们是在“01_CentralFunctions”文件夹内的“03_Manual”中创建的。在下文中，重点介绍设置“LAF_ManualOperation1_1280x800”。

此画面包含不同区域的选项卡（通过使用脚本获取区域编号来访问相应的文本列表）和一个将打开“LAF_ActvInst1_1280x800”画面的主画面窗口。

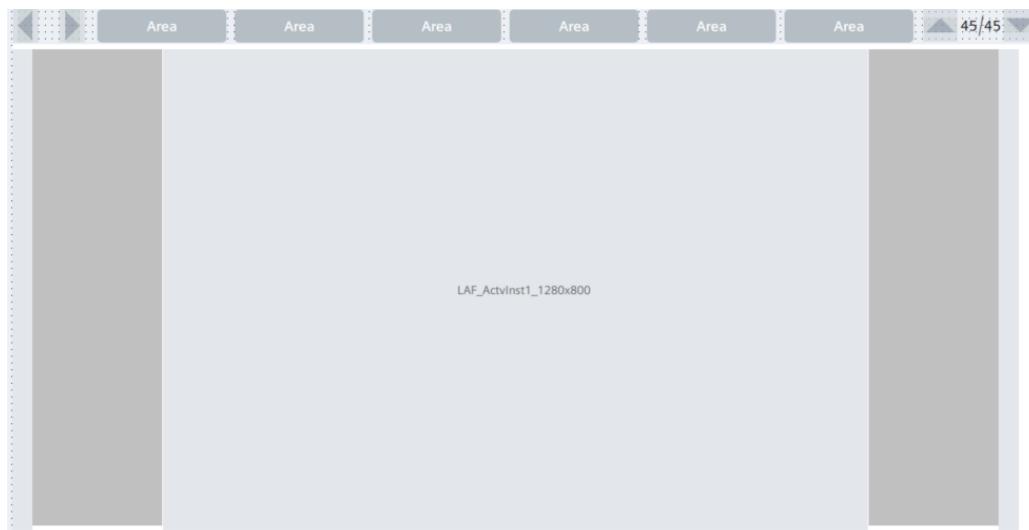


图 8-16 “LAF_ManualOperation1_1280x800”画面

- “LAF_ActvInst1_1280x800”：此画面包含手动操作行的面板，位于它们应出现的位置。每个面板都应组态为通过接口连接到 HMI 变量，这些变量将在手动行中显示所需信息。在下一张图片中，是 line0 的接口组态示例（这将是画面中出现的第一个，它在 HMI 变量表中连接到 PLC 中 DB 的手动操作行 1）：

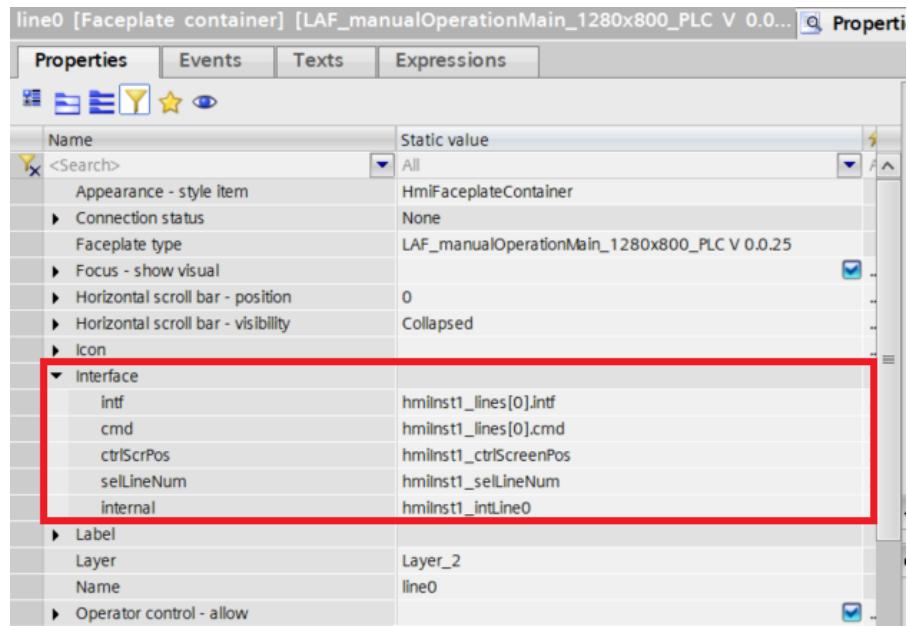


图 8-17 手动操作行 0 面板的接口组态

最后一步是用于渲染手动操作行的 Json 文件。如果没有这些文件，面板将在画面中显示为空白。必填信息包括行名称、单元、终端开关数量、按钮颜色、终端开关颜色和终端开关名称。

每个手动操作行的 json 文件必须存储在 HMI 变量 “manualOperationFilePath” 中指定的文件夹中，在本框架演示中为 “D:\JsonFiles\” 。在此， json 文件必须命名为 “<X>_<Y>_<Z>.json” ，其中 X 是 EM 编号， Y 是行号， Z 是语言代码。 json 文件的结构必须遵循下图中的示例（终端传感器的数量最多可以为 8 个右端传感器和 8 个左端传感器）：

```
3020_1_1033.json
D: > JsonFiles > 3020_1_1033.json > ...
1   [{"lEndPosNum": "1", "rEndPosNum": "1", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9",
2    "Line": "1", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "2", "rEndPosNum": "2", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "2", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "3", "rEndPosNum": "3", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "3", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "4", "rEndPosNum": "4", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "4", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "5", "rEndPosNum": "5", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "5", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "6", "rEndPosNum": "6", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "6", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "7", "rEndPosNum": "7", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "7", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "8", "rEndPosNum": "8", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "8", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "9", "rEndPosNum": "9", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "9", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "10", "rEndPosNum": "10", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "10", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "11", "rEndPosNum": "11", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "11", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "12", "rEndPosNum": "12", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "12", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "13", "rEndPosNum": "13", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "13", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "14", "rEndPosNum": "14", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "14", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "15", "rEndPosNum": "15", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "15", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "16", "rEndPosNum": "16", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "16", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "17", "rEndPosNum": "17", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "17", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "18", "rEndPosNum": "18", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "18", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "19", "rEndPosNum": "19", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "19", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "20", "rEndPosNum": "20", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "20", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}, {"lEndPosNum": "21", "rEndPosNum": "21", "unit": "m/s", "lBtnC": "9", "rBtnC": "9", "lEndC": "9", "rEndC": "9", "Line": "21", "Sym": "U3Em2_Cylinder", "Elec": "ELEC: U3Em2 Cylinder", "Mech": "MECH: U3Em2 Cylinder"}]
```

图 8-18 软件单元 3 (3020_1_1033.json) 中 EM2 的手动操作行 1 的 Json 文件结构示例

注

TIA Portal 项目包含一个 *.json 文件帮助程序（Excel 表格），用于定义 json 文件。有关详细信息，请参见文件夹 TIAPortalProjectPath/UserFiles/ManualOperationLines。

8.5. 消息

消息有三个子菜单。前两个处理当前报警/消息查看器，第二个打开历史查看器，而第三个子导航打开查看器以进行审计跟踪。

8.5.1. 报警

当前的报警和警告可以通过主导航的“消息”按钮显示。要查看报警历史记录，请使用子导航中的相应按钮。消息使用 WinCC Unified “报警控制” 显示。

使用“报警历史记录”按钮，可以显示归档的消息。还会显示系统诊断。

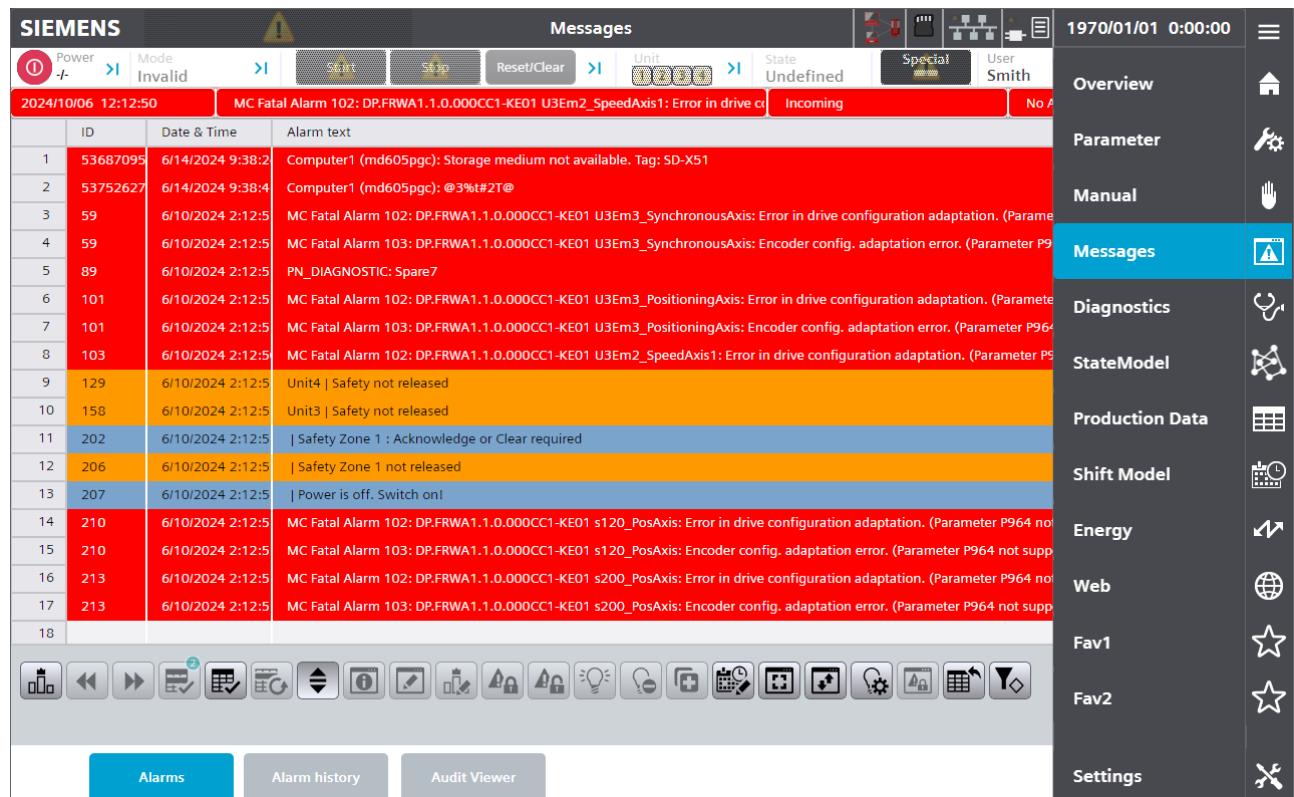


图 8-19 画面消息

8.5.2. 审计

当必须跟踪操作员的行为时，例如，为了遵守 FDA 指南 21、CRF Part11 或欧盟法规 178/2002 等法规，应使用 Audit Trail。

简介

“组态 GMP 合规”是指按照“良好生产规范”创建项目。这些要求在 FDA 规则 21 CFR Part 11 中规定。FDA 表示美国食品药品监管局。此外，EMA（欧洲药品管理局）法规 178/2002 的 Eudralex 第 4 卷附录 1 也适用。

GMP 合规和 Audit Trail

WinCC Unified 提供“审计”选项，用于实施 GMP 合规性。使用审计选项，可以启用“组态 GMP 合规”功能。

直接在 HMI 设备的运行系统设置中启用“组态 GMP 合规”功能。然后，将 GMP 合规功能添加到 WinCC Unified。

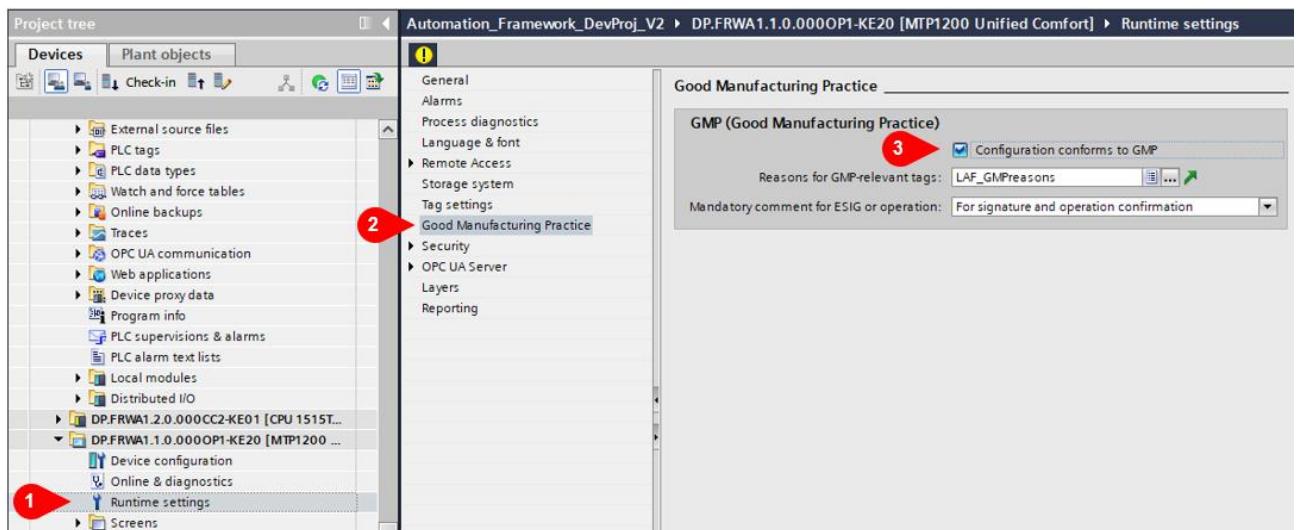


图 8-20 启用良好生产规范

这些功能包括:

- 电子签名。
- 将标签标记为“GMP 合规”的选项。
- 可以通过选择文本列表来预定义备注的建议以及 GMP 合规标签更改的典型原因。
- 自动识别 GMP 合规标签的重大变化。
- 生成和存储相关变更的电子记录 - Audit Trail。
- 记录相关用户操作的系统功能 - 电子记录。
- 记录包含校验和的日志数据。
- 用于打印记录的更改的 Audit Trail 记录。

对标记对象的执行或更改保存在一个特殊的日志中，即“AuditTrail”。GMP 合规组态意味着 HMI 设备具有电子生产数据记录功能。

日志记录范围:

下列过程是与审计相关的过程，将自动保存到 Audit Trail 中：

- 运行系统序列
 - 运行系统启动和运行系统停止。
 - 项目信息：组态环境、设备和当前运行组态的版本和项目名称。
 - 有源不间断电源 (UPS) 的电压供应故障。
 - 用户更改 GMP 合规的变量的值
- 用户管理
 - 用户登录和注销。
 - 无效登录尝试。
 - 用户管理的导入。
 - 用户管理的更改。
- 对于 GMP 合规配方：
 - 更改和创建配方数据记录后存储。
 - 配方数据记录传入或传出 PLC。
 - 对于配方变量：更改变量值与 PLC 的同步设置（“离线” / “在线”）。
- NotifyUserAction
 - 使用系统功能“InsertElectronicRecord”来记录 Audit Trail 不会自动记录的用户操作。
 - 可以为画面对象事件组态此系统功能。

Audit Viewer

可以在 TIA Portal 工具箱中找到 Audit Viewer。

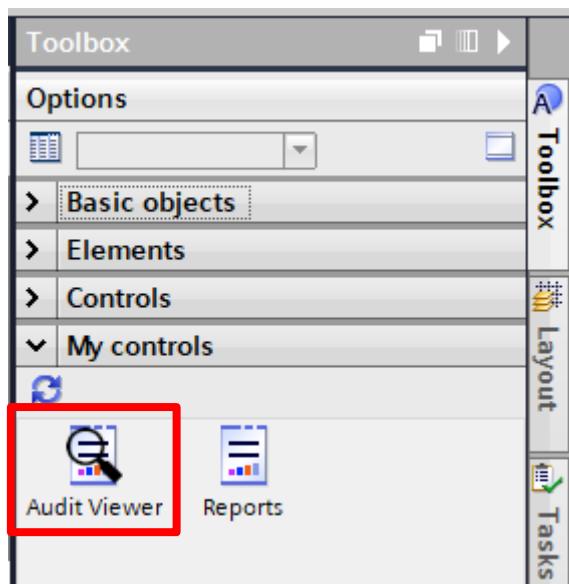


图 8-21 工具箱 – Audit Viewer

Audit Viewer 已在 HMI 的消息（主导航）和审计查看器（子导航）下实现。

要更新显示的内容，必须单击更新按钮 - 内容不会自动更新。

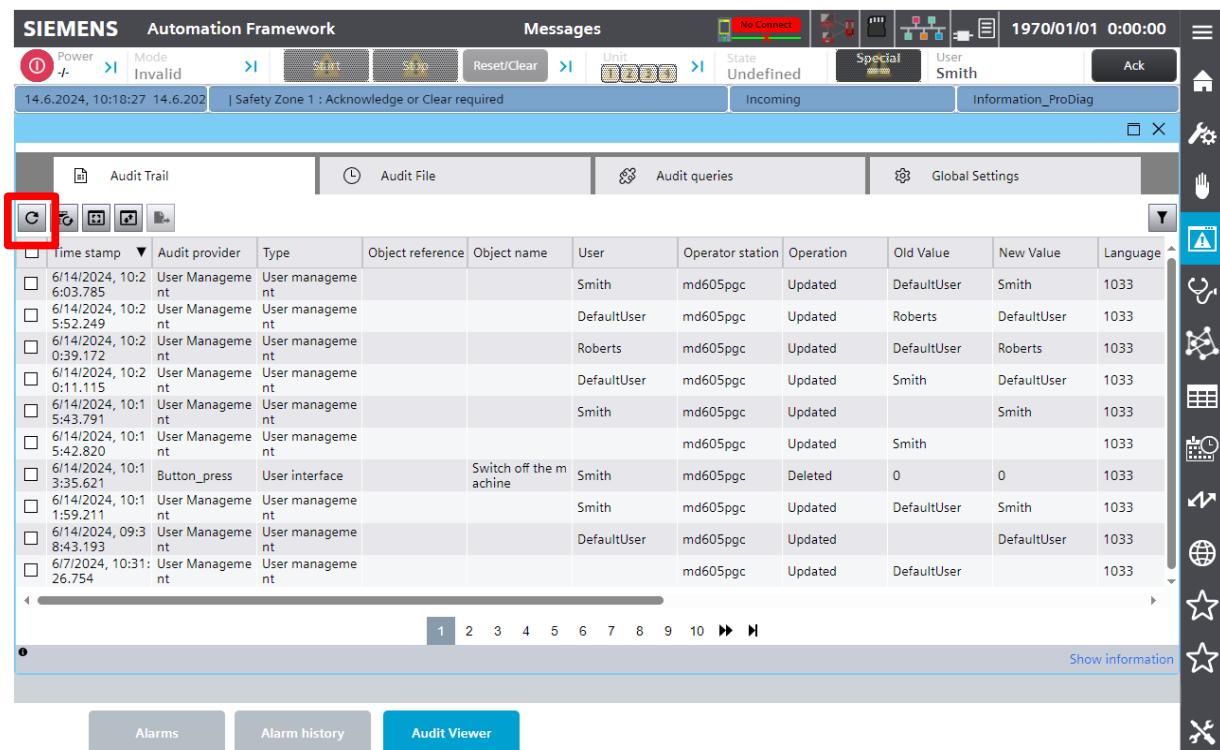


图 8-22 Audit Viewer

“InsertElectronicRecord” 的示例实现

此系统功能用于在 Audit Trail 中记录 GMP 设置不会自动记录的用户操作。也可以使用此系统功能，要求用户对操作员执行的操作进行确认、输入电子签名和备注。

“InsertElectronicRecord” 函数在 Screens/00_ScreenLayout/PopUps/Template_PowerStatus 中实现，带有 OFF 按钮的 PRESS 事件。

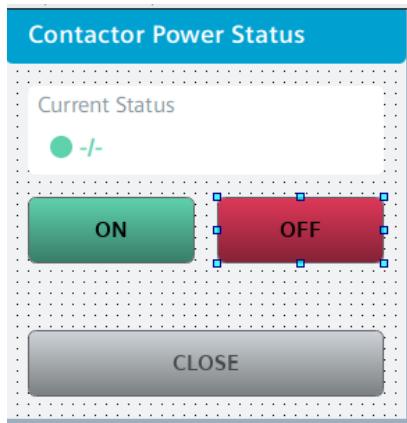


图 8-23 InsertElectronicRecord 功能



有关详细信息，请查看 WinCC Unified 文档（第 19.1 章）：

<https://support.industry.siemens.com/cs/ww/en/view/109828368>。

此外，还提供应用示例“使用 WinCC 实现 GMP 合规组态”：

<https://support.industry.siemens.com/cs/ww/en/view/109744244>。

8.6. 诊断

AF 中正在实施各种预定义的、即用型的诊断画面。以下章节介绍不同的诊断实现。

8.6.1. 系统诊断

AF 在 HMI 中提供集成系统诊断功能，使调试工程师、操作员、维护和服务人员能够快速诊断多个 PLC 和连接的 PROFINET IOSystems。

要显示“系统诊断”画面，请导航到 HMI 中的“诊断”(1) 和“系统诊断”(2)。WinCC Unified 对象“系统诊断控制”用于通过交通信号灯 SVG 显示多个 PLC 的诊断状态。诊断状态包含相关 PLC 的总体状态。合并状态始终是所有 PLC 中最差的状态。“诊断视图”显示所选 PLC 的诊断缓冲区和诊断事件，其中包括事件文本和时间戳。使用按钮(3)更新和选择不同的诊断事件，以及显示诊断事件的详细信息。

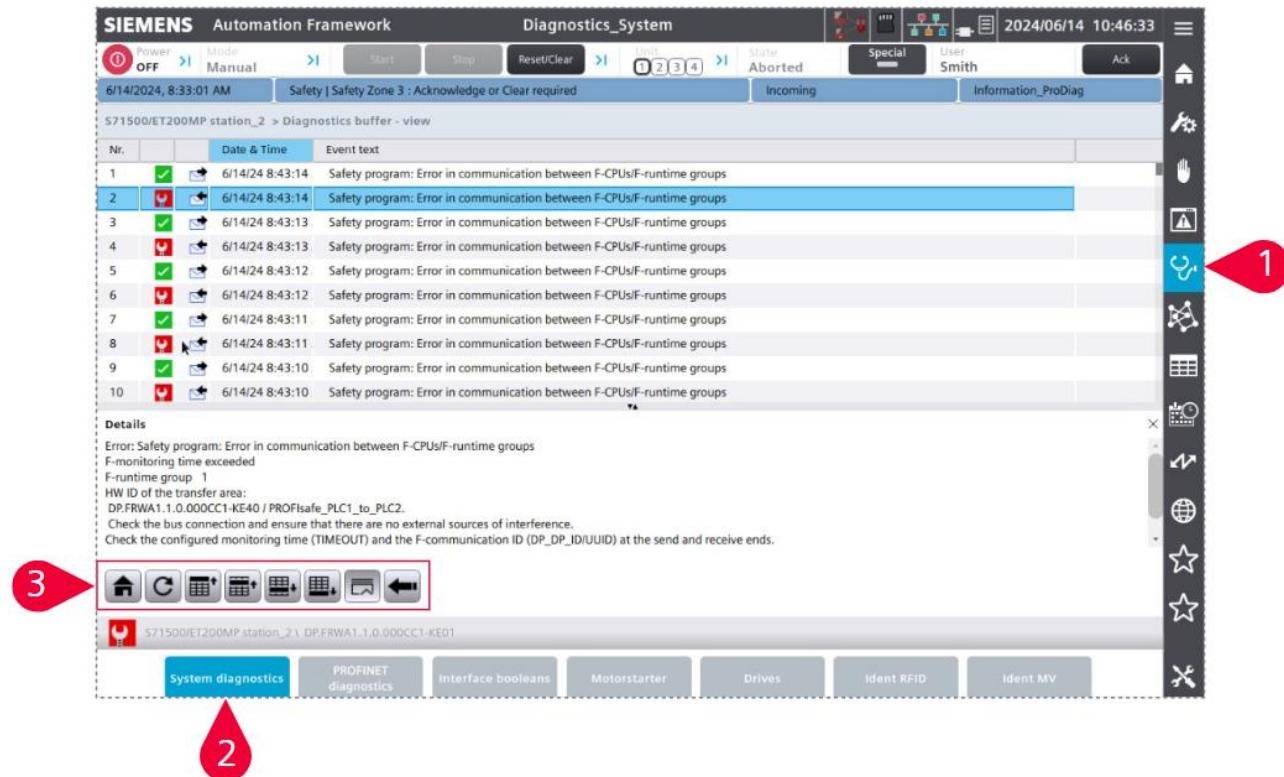


图 8-24 系统诊断（诊断视图）

注

在可用性方面，建议通过“PROFINET 诊断”的“矩阵视图”在多个 PLC 之间导航（详细信息参见 [PROFINET 诊断](#)）。



有关 WinCC 控制对象“系统诊断控制”的详细文档，请参见 TIA Portal 中的在线文档或 WinCC Unified 手册：

<https://support.industry.siemens.com/cs/ww/en/view/109828368>

8.6.2. PROFINET 诊断

此外，AF 还提供了 PLC 及其在 HMI 中 PROFINET IOSystem 的较低级别的硬件组件的详细状态诊断信息。要显示“PROFINET 诊断”画面，请导航到 HMI 中的“诊断”(1) 和“PROFINET 诊断”(2)。

该功能是通过 WinCC Unified 对象“系统诊断控制”的“分布式 IO 视图”实现的。每个硬件组件都显示为一个图块。通过单击相关图块(3)可检索有关诊断状态的详细信息。在各个硬件组件之间导航基于网络拓扑，也可以通过单击图块来实现。

如果只组态了一个带有 PROFINET IO 系统的 PLC，则“PROFINET 诊断”画面将默认显示“分布式 IO 视图”。否则，WinCC Unified 对象“系统诊断控制”在运行期间将切换为“矩阵视图”。“矩阵视图”也可以通过“起始页”按钮(4)进行检索，该按钮允许在多个 PLC 和 PROFINET IO 系统之间切换，非常人性化。

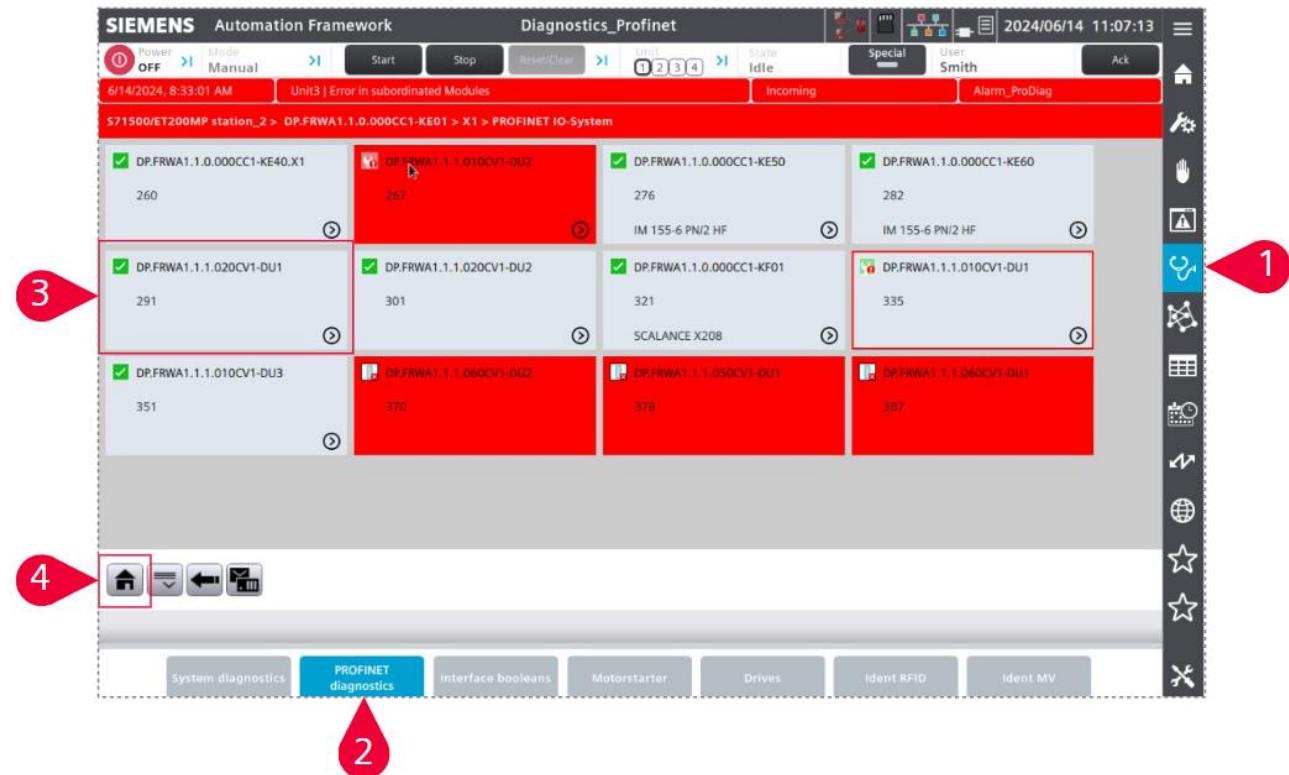


图 8-25 PROFINET 诊断（分布式 IO 视图）

8.6.3. 接口 Booleans

在 AF 中，可以监视各种布尔过程信号（例如，光栅信号）的状态，以便进行诊断。这些信号在工程组态过程中被选中，在 PLC 中组态一次，然后在操作过程中，选定的信号可以在 HMI 中显示。

为此，“CallInterfaceSignals”内部的“05_Diagnostics”文件夹中的编程块是 FB “LSicar_InterfaceSignalsBool”，必须连接每个监视信号。FB 在 HMI 接口的信号中聚合，并显示在 HMI 上。

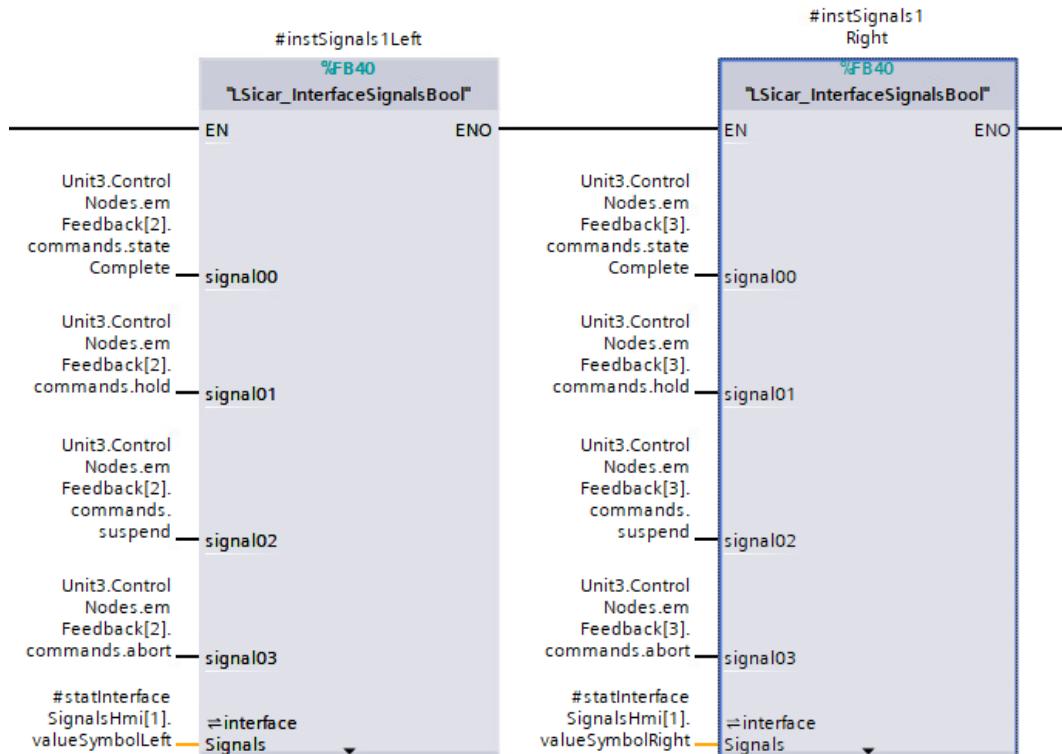


图 8-26 接口信号调用。每个块最多可以传输 16 个信号

显示的变量页面将响应第三导航中的选定页面。有 16 个可用页面，每个页面中有 32 个变量。这 32 个变量进一步分为两个部分（左侧和右侧部分）。总之，最多可以组态 512 个布尔值，执行监视功能。

可以使用左上角的铅笔图标定义每个部分的标题文本。这可以通过 HMI 进行设置（但在 PLC 停止/启动后再次被覆盖），或者可以在工程组态过程中设置初始值（在 PLC 停止/启动后保持）。布尔值的状态通过图标显示，其中 TRUE = 绿色，FALSE = 灰色。此外，HMI 会自动从“LSicar_InterfaceSignalsBool”FB 导入变量名称。

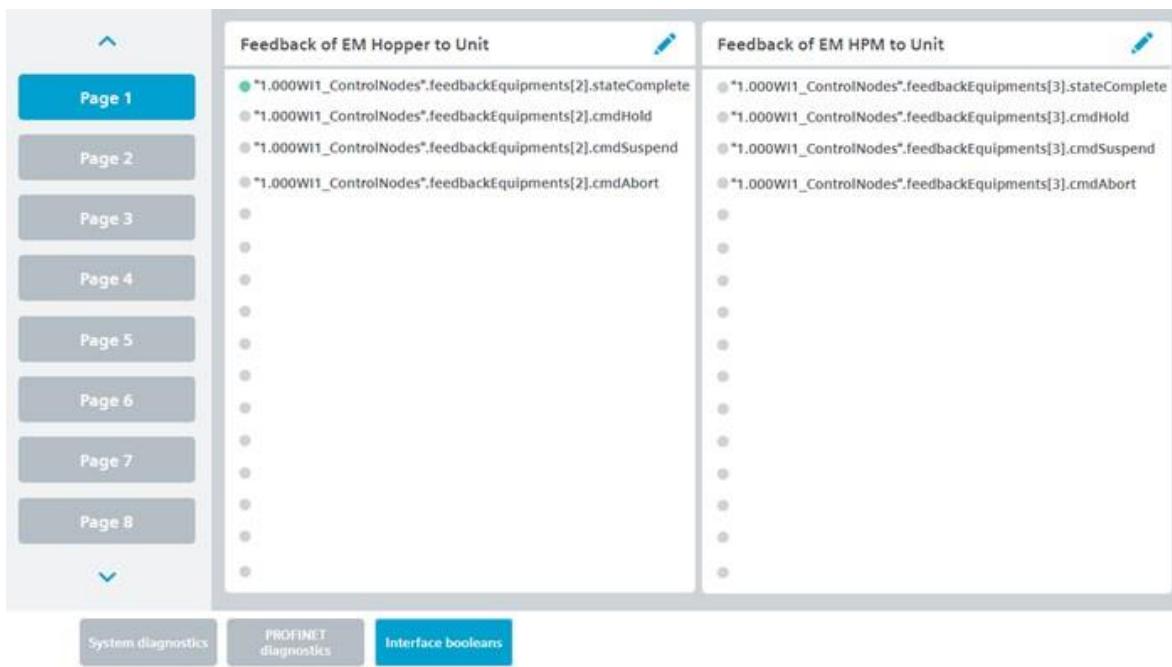


图 8-27 接口 Booleans 的诊断画面

例如，EM “EM Hopper” 和 “EM HPM” 对单元的反馈在第 1 页定义（图 8-27）。此外，在 PLC 代码中组态了相应的文件头。HMI 上显示的信号将在 PLC 的 “CallInterfaceSignals” 块中定义。它可以在作为诊断一部分的中央功能的文件夹中找到。“CallInterfaceSignals” 在 “CallDiagnostics” 块中被调用。在 “CallInterfaceSignals” 块中，“LSiCar_InterfaceSignalsBool” 被调用 32 次以读出 512 个信号。

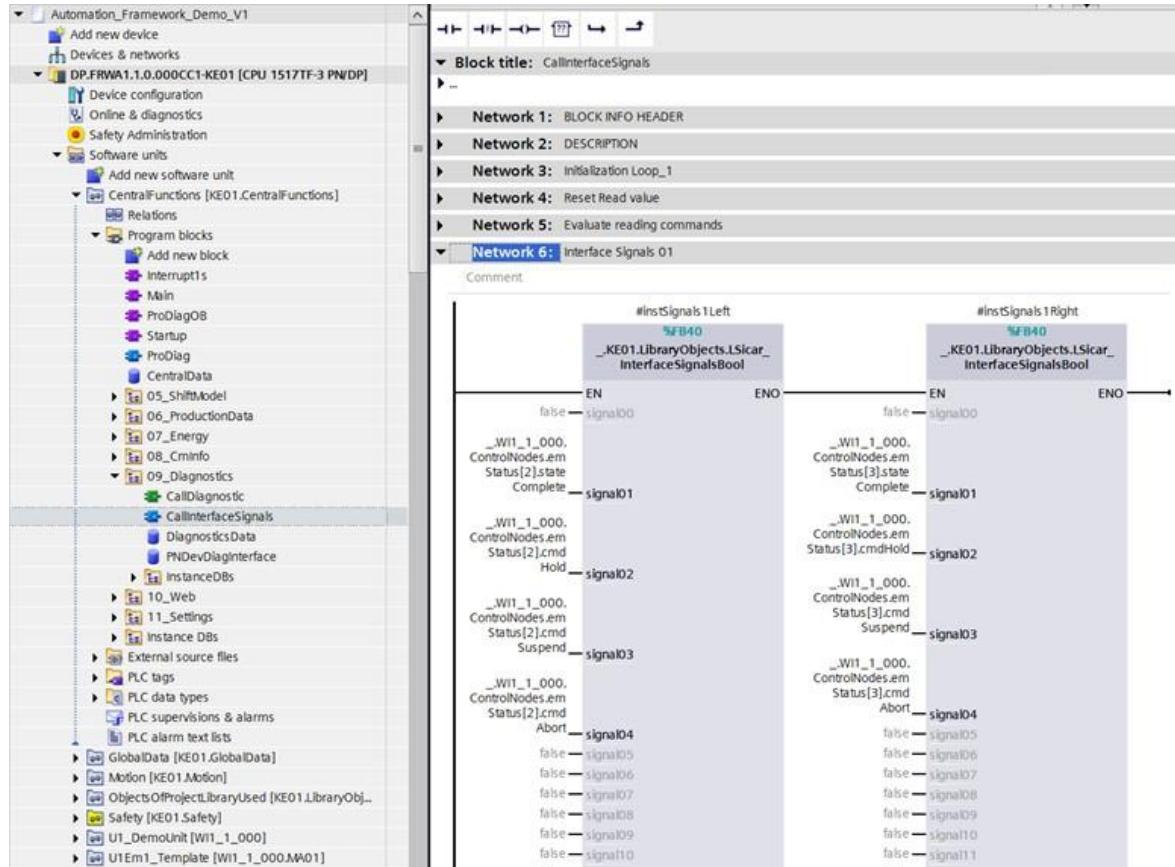


图 8-28 调用函数块，可以读出布尔变量的值和名称

函数块的结构如下：在程序段 2 到 4 中，评估 HMI 上是否发生了页面更改以及当前加载了 HMI 的哪个页面。此信息用于动态读取变量的值和名称。

在 HMI 上更改页面时，变量的名称仅读取一次。当前页面上的 32 个变量的值将连续读取。但是，只有当前页面上可见的值才会被读取。为此，使用了 FB 的接口 “interfaceSignals”。用于读取变量的类型和 FB 在程序段中以与 HMI 上的映像相同的方式进行排序。每个页面都有自己的程序段。在这个程序段中，有两个块：一个用于监视显示器的左侧部分，另一个用于监视显示器的右侧部分。数据结构 “statInterfaceSignalsHmi” 不仅用于控制变量名称和值的读取和写入，还用于存储值和名称以及头文件文本。它还用于在 HMI 和 PLC 之间交换数据。

13	statInterfaceSignalsHmi	Array[0..16] of "typeInterfaceSignalsHmi"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
14	► statInterfaceSignalsHmi[0]	"typeInterfaceSignalsHmi"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
15	► statInterfaceSignalsHmi[1]	"typeInterfaceSignalsHmi"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
16	► valueSymbolLeft	"LSicar_typeInterfaceSignalsBool"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
17	► readValue	Bool	false	Non-retain	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
18	► readSymbolicName	Bool	false	Non-retain	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
19	► freeDescriptionText	WString[40]	WSTRING#“Feedback of EM Hopper to Unit”	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
20	► interfaceSignalBool	Array[1..16] of "LSicar_typeInterfaceSig...		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
21	► interfaceSignalBool[0]	"LSicar_typeInterfaceSignals"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
22	► value	Bool	false	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
23	► symbolName	WString[60]	WSTRING#“	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
24	► interfaceSignalBool[2]	"LSicar_typeInterfaceSignals"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
25	► interfaceSignalBool[3]	"LSicar_typeInterfaceSignals"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
26	► interfaceSignalBool[4]	"LSicar_typeInterfaceSignals"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
27	► interfaceSignalBool[5]	"LSicar_typeInterfaceSignals"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
28	► interfaceSignalBool[6]	"LSicar_typeInterfaceSignals"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
29	► interfaceSignalBool[7]	"LSicar_typeInterfaceSignals"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
30	► interfaceSignalBool[8]	"LSicar_typeInterfaceSignals"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
31	► interfaceSignalBool[9]	"LSicar_typeInterfaceSignals"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
32	► interfaceSignalBool[10]	"LSicar_typeInterfaceSignals"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
33	► interfaceSignalBool[11]	"LSicar_typeInterfaceSignals"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
34	► interfaceSignalBool[12]	"LSicar_typeInterfaceSignals"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
35	► interfaceSignalBool[13]	"LSicar_typeInterfaceSignals"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
36	► interfaceSignalBool[14]	"LSicar_typeInterfaceSignals"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
37	► interfaceSignalBool[15]	"LSicar_typeInterfaceSignals"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
38	► interfaceSignalBool[16]	"LSicar_typeInterfaceSignals"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
39	► valueSymbolRight	"LSicar_typeInterfaceSignalsBool"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
40	► readValue	Bool	false	Non-retain	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
41	► readSymbolicName	Bool	false	Non-retain	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
42	► freeDescriptionText	WString[40]	WSTRING#“Feedback of EM HPM to Unit”	Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
43	► interfaceSignalBool	Array[1..16] of "LSicar_typeInterfaceSig...		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
44	► statInterfaceSignalsHmi[2]	"typeInterfaceSignalsHmi"		Non-retain	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

图 8-29 HMI 接口为接口 Booleans

显示的 HMI 结构 (statInterfaceSignalHmi) 是一个包含 16 个元素的数组。每个元素都包含数据类型为 “LSicar_typeInterfaceSignalsBool”的两个元素。一个用于 HMI 页面的左侧部分，另一个用于右侧部分。各个元素由块描述，名称和值由 HMI 读出。HMI 中不需要额外的工程组态，除非要读取的变量超过组态的变量，在这种情况下，必须相应地调整 HMI 中的结构。在这里，可以看到用于控制名称和值 (readValue 和 readSymbolicName) 读取过程的变量，以及名称和值本身的标头 (freeDescriptionText) 和数组。由于变量的名称是自动读出的，因此只需在此处设置标题。



有关接口 Booleans 的更多信息，请参见以下文档：

[参考“LSicar_InterfaceSignalsBool”](#)

8.6.4. 驱动器诊断

通过提供的块和画面，调试工程师、操作员、维护和服务人员可以在 HMI 中快速诊断工艺对象（轴）和 SINAMICS 驱动器。驱动器诊断包括 TO 和 SINAMICS 驱动器的状态和控制信息，以及 SINAMICS 的故障消息和警告。此外，还提供有关 SINAMICS 驱动器基本定位器（EPOS）功能定位状态的诊断信息。

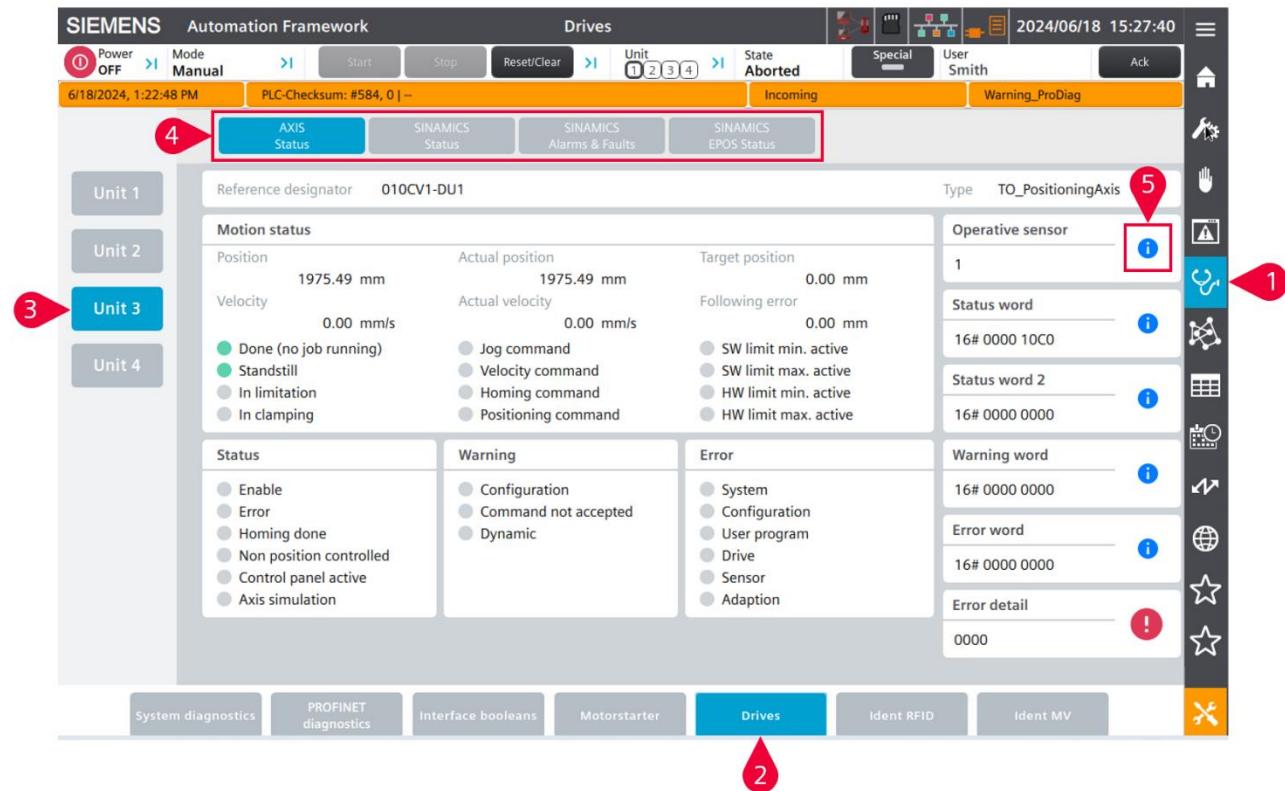


图 8-30 HMI 导航和驱动器诊断画面布局

要显示驱动器诊断画面，请导航到“诊断”(1) 和“驱动器”(2)。然后通过第三级导航(3)选择相关驱动器。通过导航栏(4)，每个驱动器诊断的基本信息都显示在主画面中。详细的状态信息可通过弹出窗口(5)进行检索。

8.6.4.1. 概览

驱动器诊断由三个主要部分组成。数据块用于存储组态数据，并用作 HMI 的接口。函数块读取诊断数据并将其编辑为可视化格式。画面以清晰、结构化的布局显示 HMI 中的诊断数据。每个 HMI 画面都连接到 PLC 中的一个函数块。下表提供了所有块和画面的概述：

块	功能	画面
DB “DriveDiagConfig”	用于驱动器诊断的所有驱动器对象的组态数据数组。	-
DB “DriveDiagInterfaceHmi”	用于驱动器诊断的 HMI 接口。	-
FC “LAF_SetDriveObjectConfig” (可选)	该块根据 PLC 启动期间的参数化索引和输入值设置组态数据数组中驱动器对象的组态数据。	-

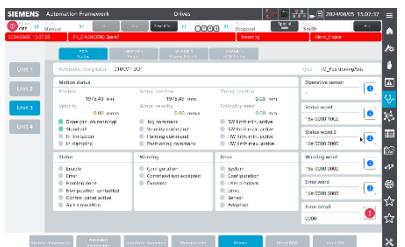
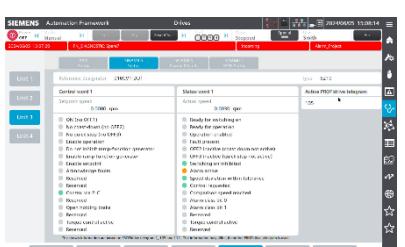
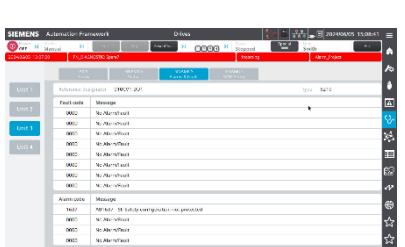
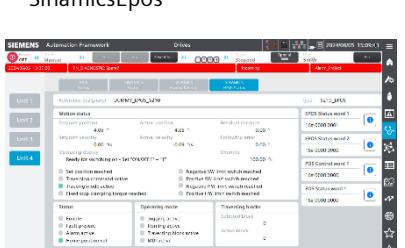
块	功能	画面
FB “LAF_GetAxisStatus”	该块根据组态的工艺对象和操作员在 HMI 中选定的驱动对象索引读取轴的诊断信息。	
FB “LAF_GetSinaStatus”	该块通过非周期性通信从 SINAMICS 驱动器读取 PROFIdrive 报文状态字 1 (ZSW1) 和控制字 1 (STW1) 以及设定值和实际速度值，具体取决于组态的 SINAMICS 类型和操作员在 HMI 中选择的驱动器对象索引。	
FB “LAF_GetSinaAlarms”	该块通过非周期性通信读取 SINAMICS 驱动器的故障和报警，具体取决于组态的 SINAMICS 类型和操作员在 HMI 中选择的驱动器对象索引。	
FB “LAF_GetSinaEpos”	该块通过非周期性通信从 SINAMICS 驱动器读取 EPOS 定位状态信息以及设定值和实际值，具体取决于组态的 SINAMICS 类型和操作员在 HMI 中选择的驱动器对象索引。	

表 8-23 驱动器诊断块和画面

8.6.4.2. 架构

以“SinamicsEpos”画面和“LAF_GetSinaEpos”函数块为例，对架构进行了说明。

操作员通过 HMI 中的第三导航选择相关驱动器。通过执行此操作，设置了“DriveObjectIndex”。 “DriveObjectIndex” 和活动的“SinamicsEpos”，画面信息 (1) 通过“DriveDiagInterfaceHMI”数据块 (2) 传输到 PLC。“LAF_GetSinaEpos”函数块 (3) 由活动的“SinamicsEpos”画面启用。此函数块根据“DriveObjectIndex”从“DriveDiagConfig”数据块 (4) 中读取组态。部分组态数据直接传输到 HMI。另一部分用于建立非周期性通信，以从 SINAMICS 驱动器读取参数。接收到的参数先调整格式，然后传输到 HMI (5)。

“LAF_SetDriveObjectConfig”功能 (A) 在 PLC 启动期间为“DriveDiagConfig”数据块中的一个驱动器诊断对象设置组态。

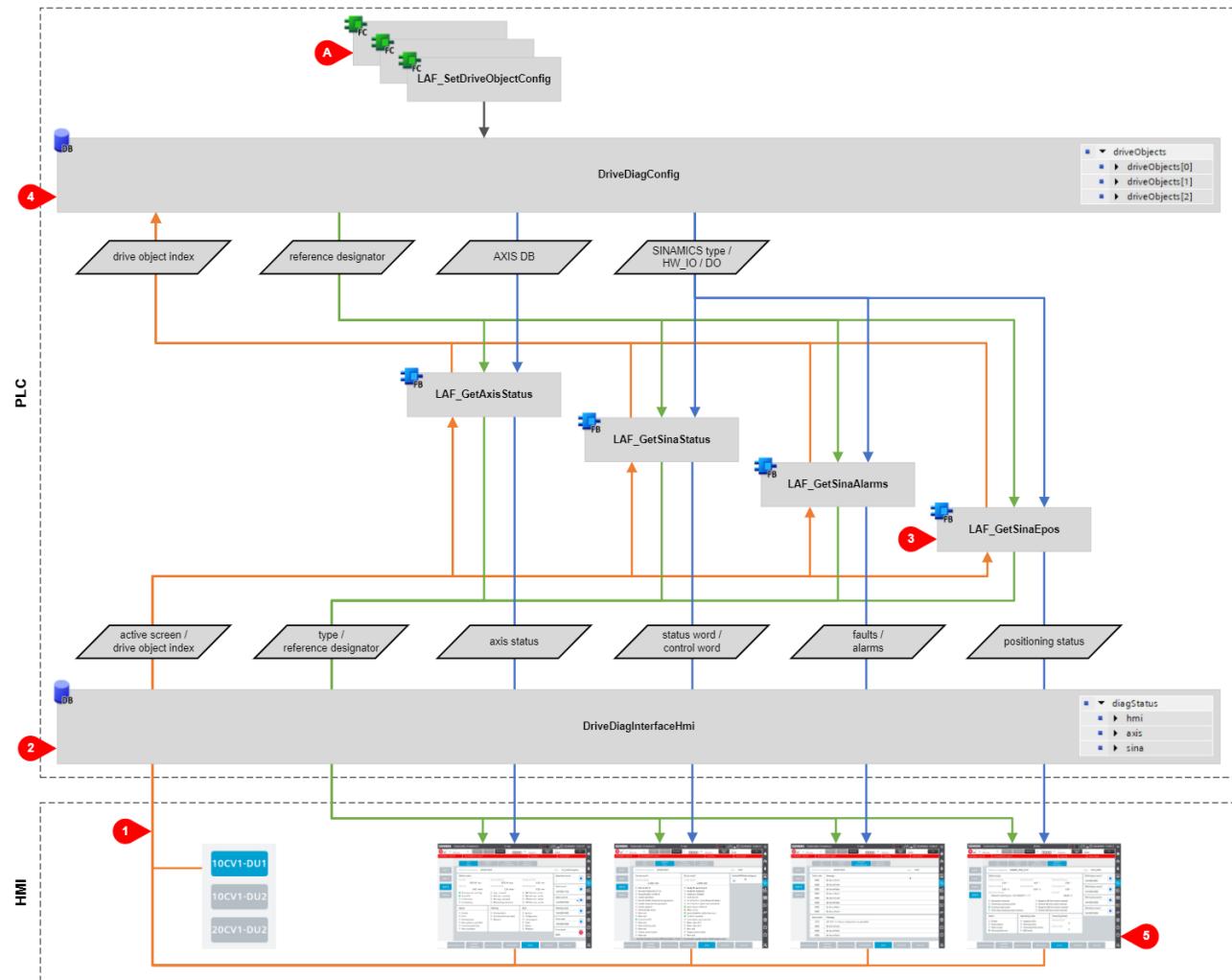


图 8-31 驱动器诊断架构的蓝图

组态数据

为了便于应用工程师操作，所有驱动器诊断对象都在同一个位置进行组态。此组态存储在“DriveDiagConfig”中。每个驱动器诊断对象都有自己的数组元素。

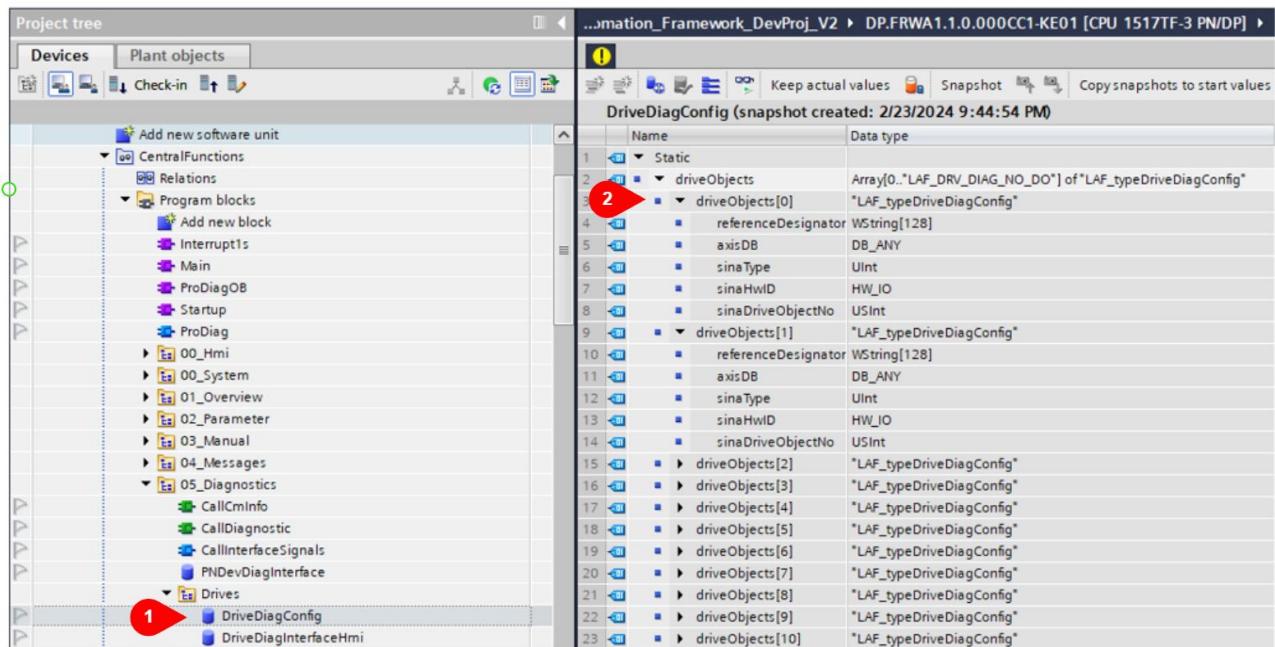


图 8-32 组态数据块“DriveDiagConfig”的结构

注

为降低系统负载，建议在 PLC 启动时使用“LAF_SetDriveObjectConfig”功能设置每个驱动器诊断对象的组态。

每个驱动器诊断对象都包含以下变量：

变量	描述
referenceDesignator	HMI 中显示的驱动器诊断对象的名称或参考指示符。
axisDB	工艺对象的 DB 编号。如果未使用 TO，则设置 axisDB = 0。 支持以下类型的 TO： <ul style="list-style-type: none">• TO_SpeedAxis• TO_PositioningAxis• TO_SynchronousAxis• TO_ExternalEncoder
sinaType	SINAMICS 驱动器类型。需要该类型才能从 SINAMICS 中读取正确的参数。支持以下类型的 SINAMICS： <ul style="list-style-type: none">• 000 = 无类型，仅 TO 诊断信息 (对于 TO_ExternalEncoder 是必选的，对于其他 TO 是可选的)• 114 = SINAMICS G120，带 TO 或 SinaSpeed• 124 = SINAMICS S120 带 TO• 125 = SINAMICS S120 带 EPOS• 220 = SINAMICS S200 带 TO• 221 = SINAMICS S200 带 EPOS• 222 = SINAMICS S210 带 TO• 223 = SINAMICS S210 带 EPOS

变量

sinaHwID

描述

PROFINET IO 设备或 PROFIBUS DP 从站的硬件识别，用于寻址 SINAMICS 驱动器。在这种情况下，硬件标识是指 PROFIdrive 报文。

sinaDriveObjectNo

PROFIdrive 设备根据轴数由一个或多个功能对象组成。这些对象中的每一个都表示轴的功能，并称为驱动器对象 (DO)。
SINAMICS 驱动器的驱动器对象编号如下：
 • SINAMICS G120: 单轴，“sinaDriveObjectNo” = 1
 • SINAMICS S120: 多轴，请参见下面的指南
 • SINAMICS S200: 单轴，“sinaDriveObjectNo” = 1
 • SINAMICS S210: 单轴，“sinaDriveObjectNo” = 1

表 8-45 一个驱动器诊断对象的组态变量

要访问驱动器对象编号，请打开 SINAMICS 的“设备组态”(1)。然后，驱动器对象编号显示在“设备概述”(2) 中的“驱动器对象编号”(3) 中。这种方式适用于每个 SINAMICS 驱动器。

此外，SINAMICS S120 多轴系统的驱动对象编号可通过控制单元 (CU) 的 PROFINET 接口 (4) 进行访问。因此，在属性 (5) 中选择“报文组态”。然后，驱动器对象编号也显示在“项目”列 (6) 中。

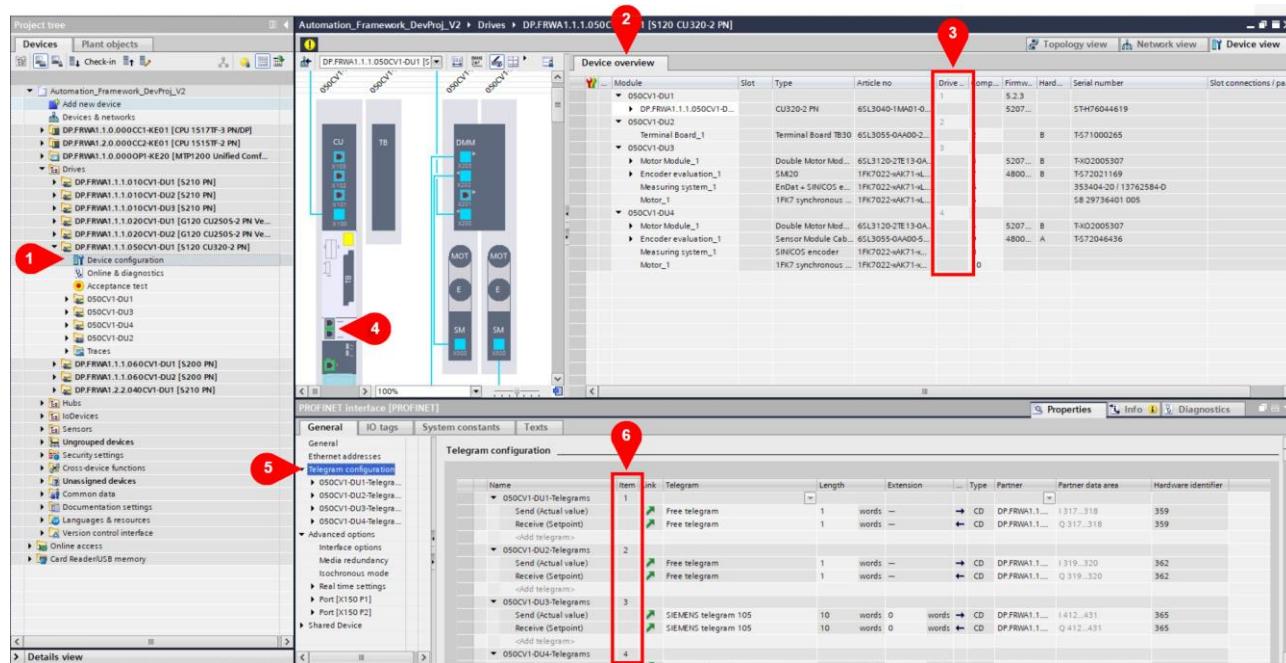


图 8-33 在 SINAMICS S120 多轴系统中访问驱动对象编号

HMI 接口

PLC 和 HMI 之间的接口是“DriveDiagInterfaceHmi”数据块(1)。该块由三个部分组成：

- HMI 控制变量以及共享诊断数据(2)。所有驱动器诊断块都使用这些变量。
- 来自“LAF_GetAxisStatus”函数块的工艺对象(3)的轴状态信息。
- Sinamics 状态信息(4)，由“LAF_GetSinaStatus”、“LAF_GetSinaAlarms”和“LAF_GetSinaEpos”函数块填充。

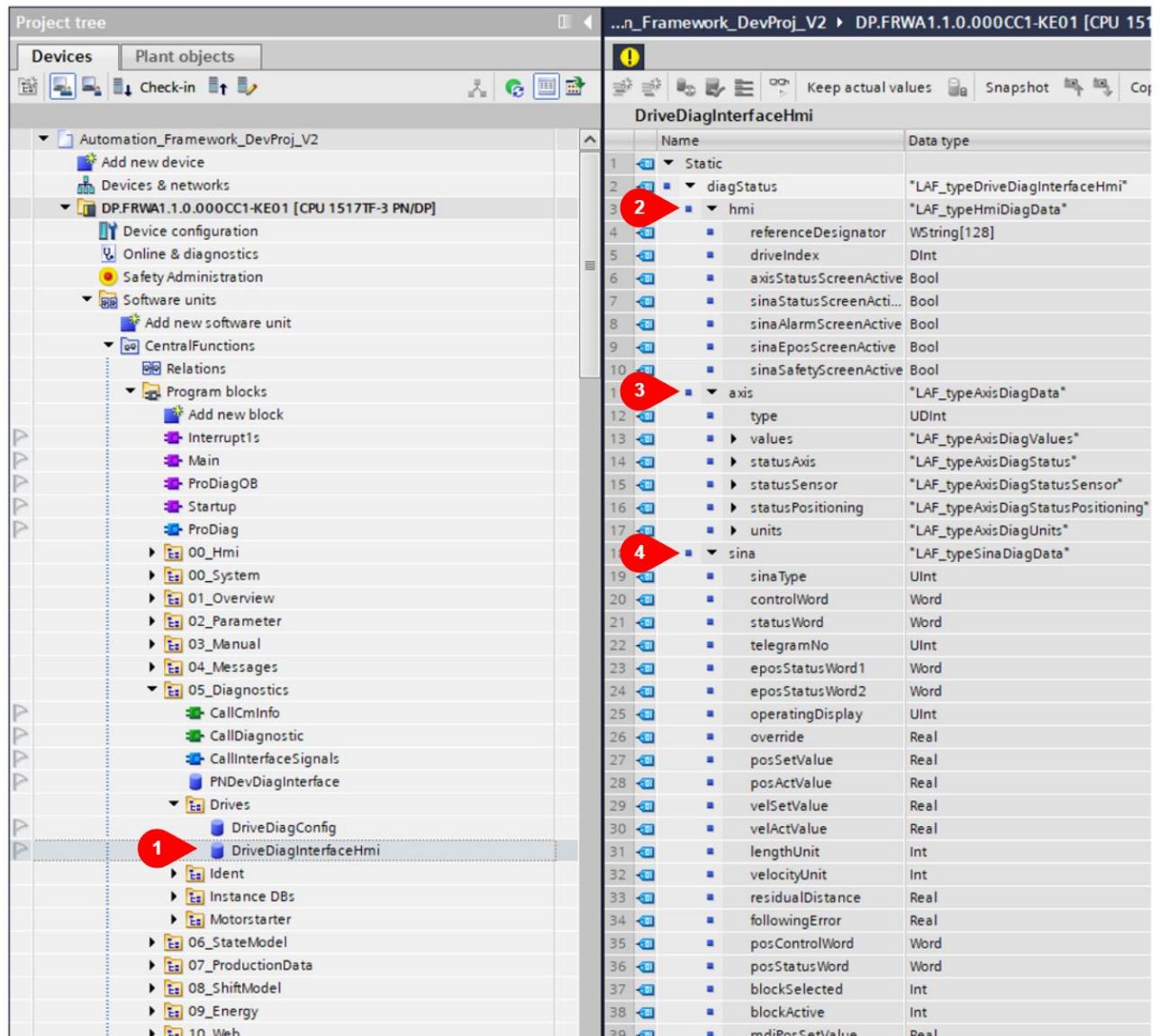


图 8-34 HMI 接口数据块“DriveDiagInterfaceHmi”的结构

注

“DriveDiagInterfaceHmi”数据块以及与 HMI 变量和画面对象的连接都在 AF 的预组态 TIA Portal 项目中实现。

PLC 函数块

PLC 功能通过四个函数块“LAF_GetAxisStatus”、“LAF_GetSinaStatus”、“LAF_GetSinaAlarms”和“LAF_GetSinaEpos”实现，它们在“CallCmInfo”函数(1)中调用，并共享两个数据块“DriveDiagConfig”和“DriveDiagInterfaceHmi”(2)。

为了减少系统负载，当 HMI 中显示相应的驱动器诊断画面时，PLC 中的每个函数块都会启用。通过在“启用”输入(3)上分配值为“FALSE”的 PLC 变量，可以从用户程序中禁用驱动器诊断函数块。此变量为“FALSE”，但不会读取驱动器诊断数据并将其传输到 HMI。

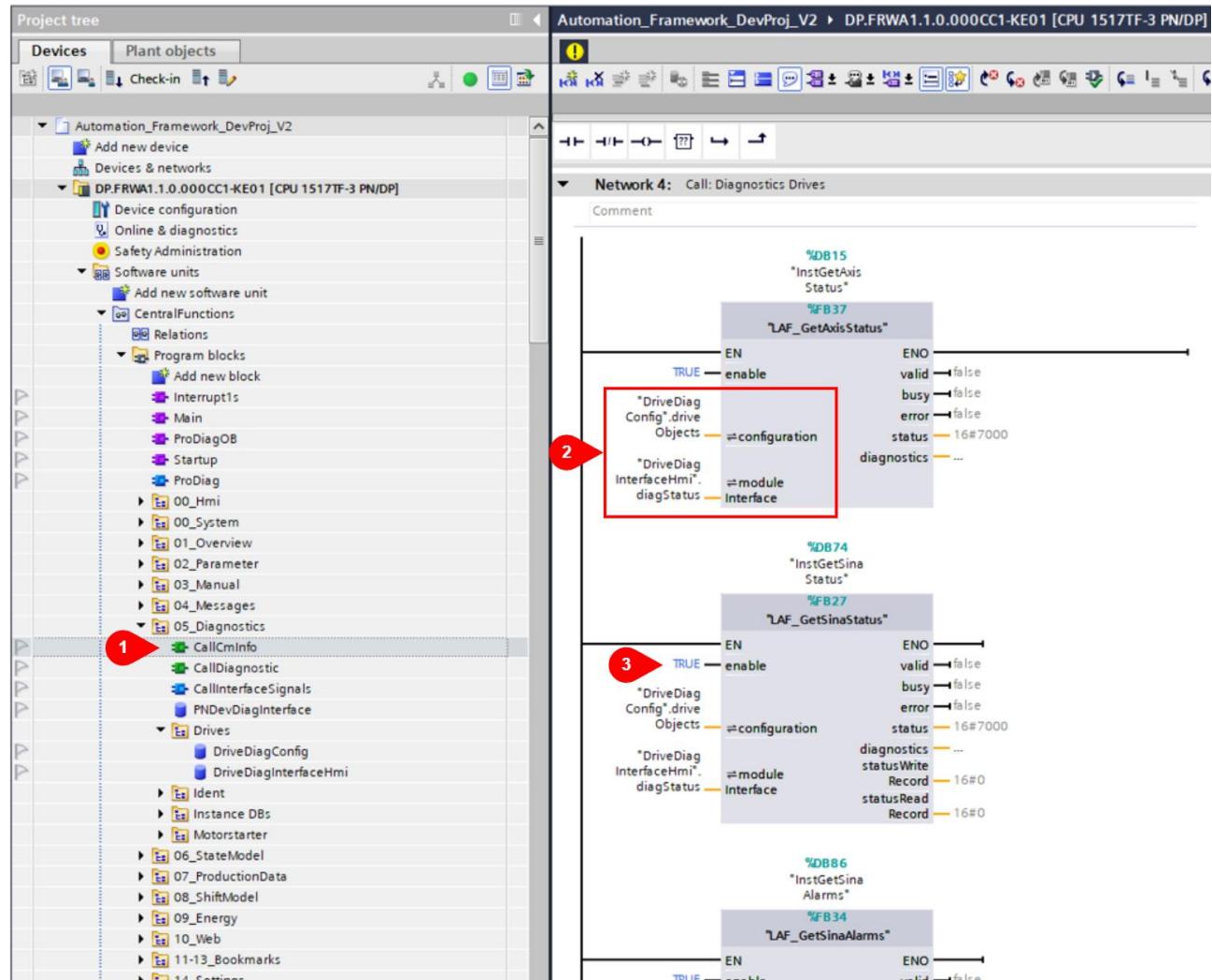


图 8-35 用于驱动器诊断的函数块调用

注

驱动器诊断函数块在“CallCmInfo”函数中调用，并连接到 AF 的预组态 TIA Portal 项目中的“DriveDiagConfig”和“DriveDiagInterfaceHmi”数据块。

8.6.4.3. 工程组态

在工程组态阶段，必须执行多个步骤来在 PLC 和 HMI 中设置驱动器诊断。为了使工程师操作方便，任务简化为在 PLC 程序的“DriveDiagConfig”数据块中设置组态，并在 HMI 中扩展第三导航。

PLC 工程组态

首先导航到“LibraryObjects”变量表 (1)。选择“用户常量”(2)，并将“LAF_DRV_DIAG_NO_DO”常量(3)的值设置为 TIA Portal 项目中的工艺对象和 SINAMICS 驱动器 (= 驱动器诊断对象) 的数量。然后编译“DriveDiagConfig”数据块。

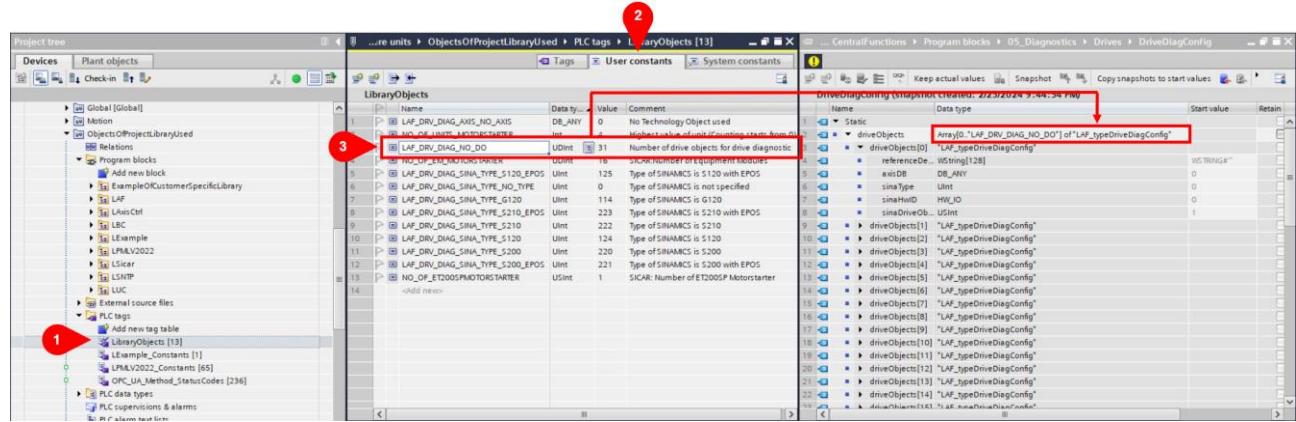


图 8-36 设置用于驱动器诊断的驱动器对象数

示例：

- 1x SINAMICS G120, 带 SinaSpeed
 - 2x SINAMICS S210 带 TO_PositioningAxis 和 TO_SynchronousAxis
 - 1x SINAMICS S210 带 EPOS
- ⇒ “LAF_DRV_DIAG_NO_DO” = 3

在此示例中，“LAF_DRV_DIAG_NO_DO”表示数组包含 4 个元素 “[0..3]”。

接下来，在软件单元“CentralFunctions”和工艺对象(2)之间创建关系(1)。

The screenshot shows the SIMATIC Manager interface with the following details:

- Project tree:** On the left, under "Devices", the "CentralFunctions" software unit is selected. A red arrow labeled "1" points to the "Relations" folder within "CentralFunctions".
- Relations table:** On the right, the "Relations" table lists 18 entries. The last two entries are highlighted with a red border and a red arrow labeled "2" points to them. These entries show relationships between "CentralFunctions" and "Technology object" (U3Em3_PositioningAxis and U3Em3_SynchronousAxis).
- Technology objects:** In the bottom-left corner of the project tree, a red box highlights the "Technology objects" section, which contains "U3Em3_HighPerfMotion" and its sub-items "U3Em3_PositioningAxis [DB20]" and "U3Em3_SynchronousAxis [DB21]."

Selected unit	Relation type	Accessible element
1 CentralFunctions	Software unit	ObjectsOfProjectLibraryUsed
2 CentralFunctions	Software unit	Safety
3 CentralFunctions	Software unit	U3_Demo_____
4 CentralFunctions	Software unit	Global
5 CentralFunctions	Software unit	U3Em1_Hopper
6 CentralFunctions	Software unit	U3Em4_HighPerfMotionEncaps
7 CentralFunctions	Software unit	U3Em0_General
8 CentralFunctions	Software unit	U3Em2_ConveyerStation
9 CentralFunctions	Software unit	U1Em1_Template_SCL
10 CentralFunctions	Software unit	
CentralFunctions	Technology object	U3Em3_PositioningAxis
CentralFunctions	Technology object	U3Em3_SynchronousAxis
13 CentralFunctions	Technology object	U3Em2_SpeedAxis
14 CentralFunctions	Technology object	U3Em4_PositioningAxis
15 CentralFunctions	Technology object	U3Em4_SynchronousAxis
16 CentralFunctions	Technology object	U4EmX_PositioningAxis1
17 CentralFunctions	Technology object	U4EmX_PositioningAxis2
18 <Add new relation>		

图 8-37 创建与工艺对象的关系

打开 OB “Startup” (1)。为“DriveDiagConfig” (3) 的每个数组元素调用“LAF_SetDriveObjectConfig”函数 (2)，并分配数组下标 (3) 以及组态数组本身 (4)。

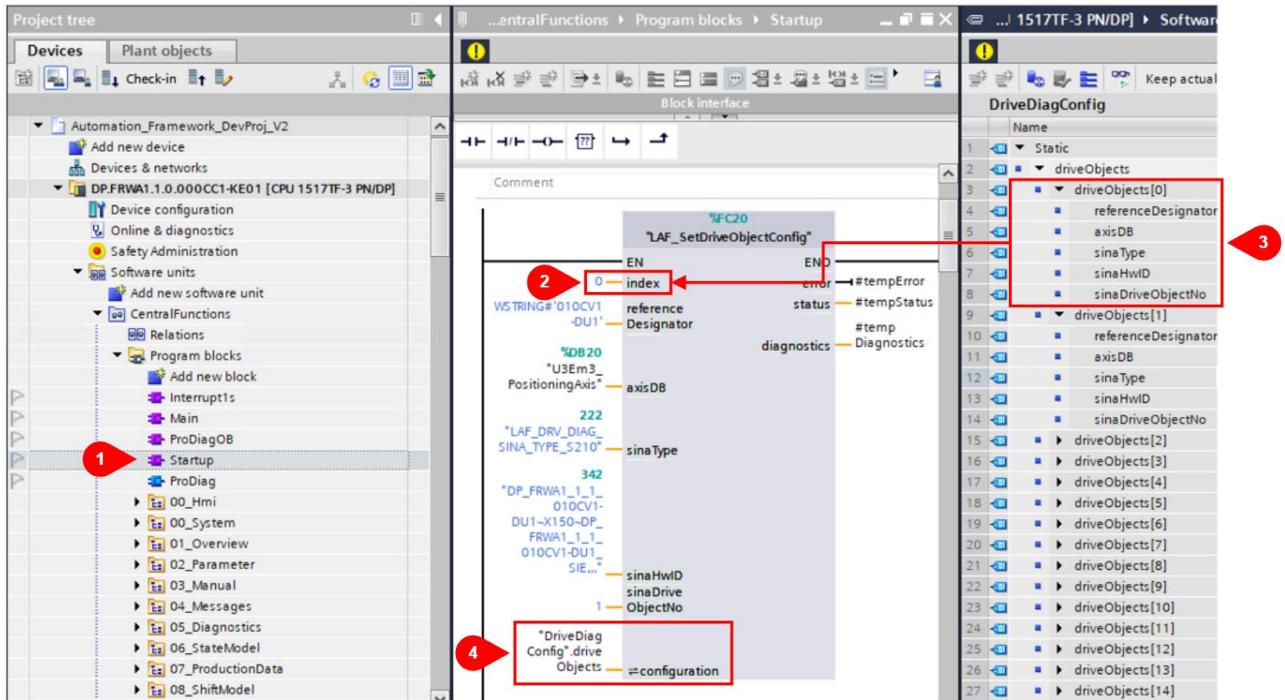


图 8-38 启动时调用“LAF_SetDriveObjectConfig”功能

接下来，通过拖放操作分配驱动器 (1) 的参考指示符和工艺对象 (2) 的 DB 以及 SINAMCS (3) 的类型。对于 SINAMICS 的类型，“LibraryObjects”变量表中的预定义常量可用。

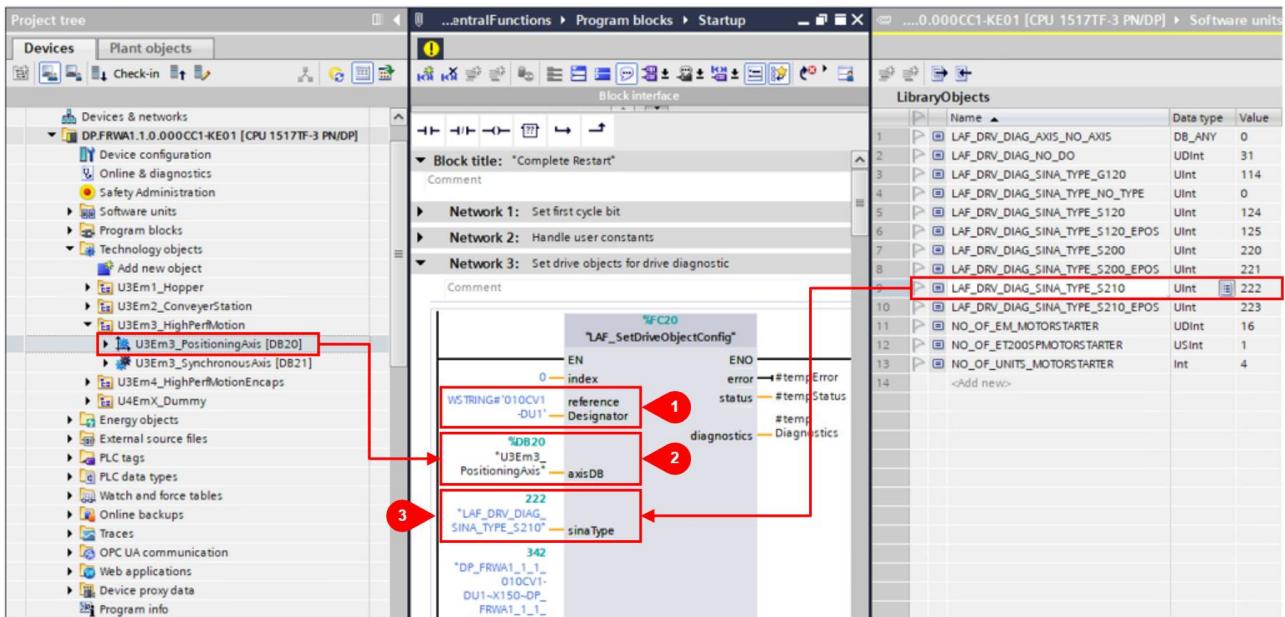


图 8-39 将 SINAMICS 的参考指示符、TO 和类型分配给“LAF_SetDriveObjectConfig”功能

然后打开相应 SINAMICS 的“设备组态”(1), 选择 PROFINET 接口(2), 然后导航到属性中的“报文组态”, 然后选择“系统常量”选项卡(3)。通过拖放将 PROFIdrive 报文(4)的系统常量分配给“LAF_SetDriveObjectConfig”功能。

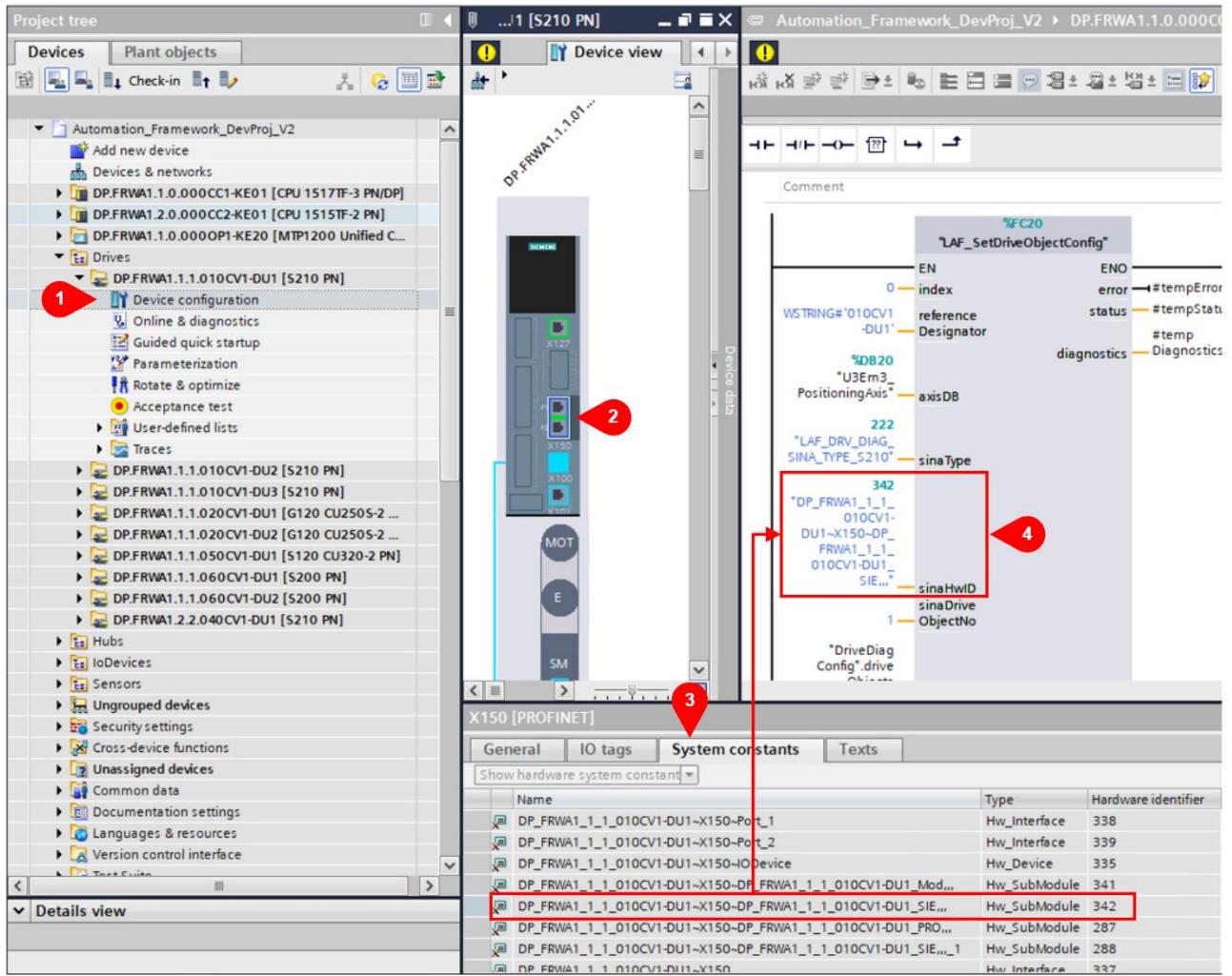


图 8-40 将 PROFIdrive 报文的系统常量分配给“LAF_SetDriveObjectConfig”功能

注

系统常量必须与 PROFIdrive 报文的硬件标识符匹配。仔细检查 SINAMICS 驱动器的“报文组态”中的硬件标识符。

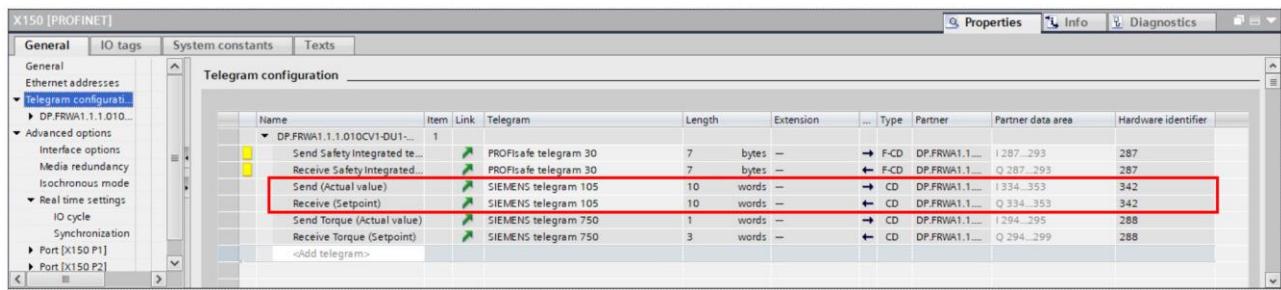


图 8-41 访问 PROFIdrive 报文的硬件标识符

最后，将驱动器对象编号分配给“LAF_SetDriveObjectConfig”功能。因此，请再次打开“设备组态”(1)，并检查“设备概述”(2)表中的驱动器对象编号。

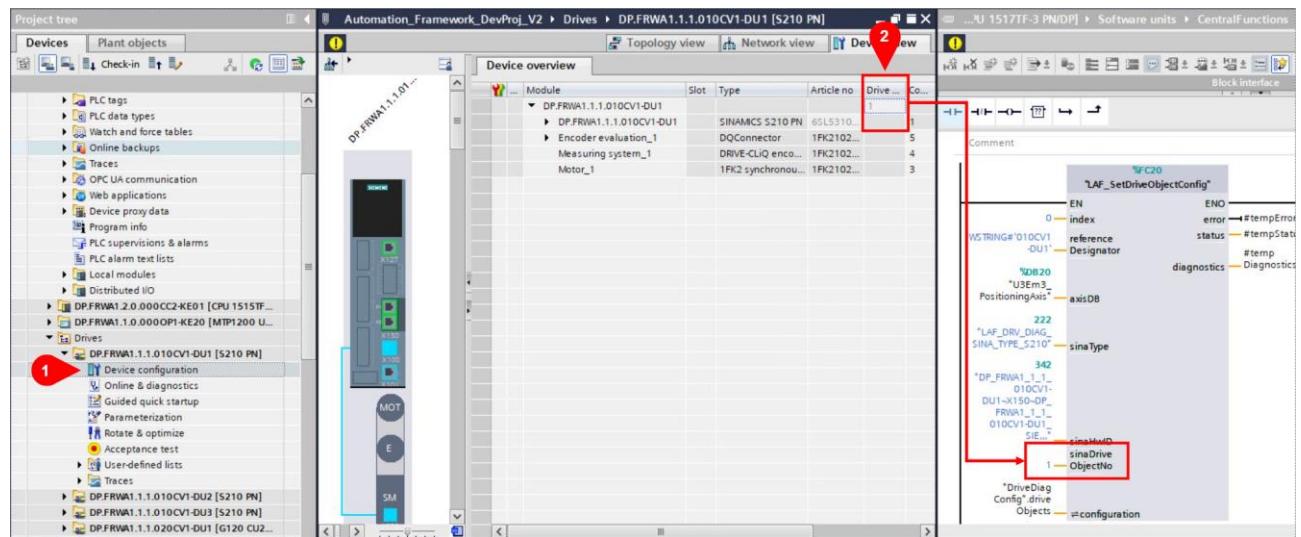


图 8-42 将驱动器对象编号分配给“LAF_SetDriveObjectConfig”功能

HMI 工程组态

用于驱动器诊断的 HMI 画面根据组态的工艺对象和 SINAMICS 驱动器进行预组态和动态化。因此，HMI 工程组态的重点简化为对第三导航的扩展。通过第三导航，操作员选择相关的驱动器诊断对象。

注 第三级导航由单元级、EM 级和 CM 级三层组成。本章介绍在控制模块级别设置驱动器诊断所需的任务。[导航](#) 中介绍了如何实现导航。

为了加快 HMI 工程组态工作流，AF 为控制模块级别 (1) 提供了一个预组态的模板画面，最多可容纳 8 个驱动器 (2)。

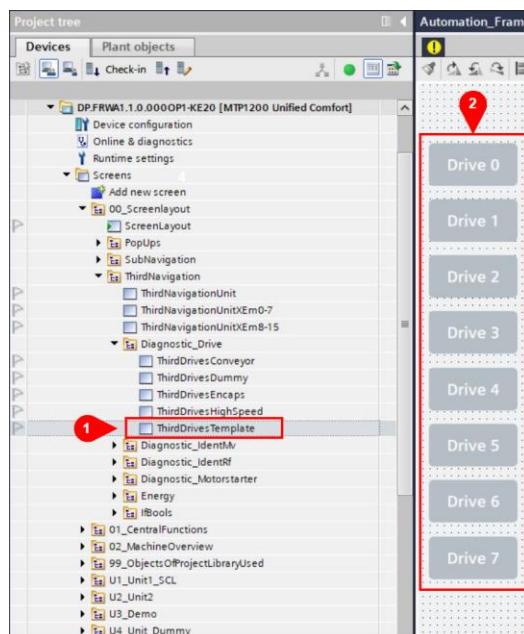


图 8-43 驱动器诊断的第三导航的模板画面

复制模板画面，根据 PLC 中对应 EM 的名称进行重命名。然后打开画面 (1) 并调整文本框的数量以及驱动控制模块 (2) 的名称或参考指示符。

接下来，将“DriveDiagInterfaceHmi”数据块中的PLC变量“DriveObjectIndex”(3)的值设置为“DriveDiagConfig”数据块中应诊断的数组元素(4)的相关下标。还可以调用脚本“Navigation.HideThirdNavigationEM”和“Navigation.HideThirdNavigationCM”以隐藏第三导航(5)。

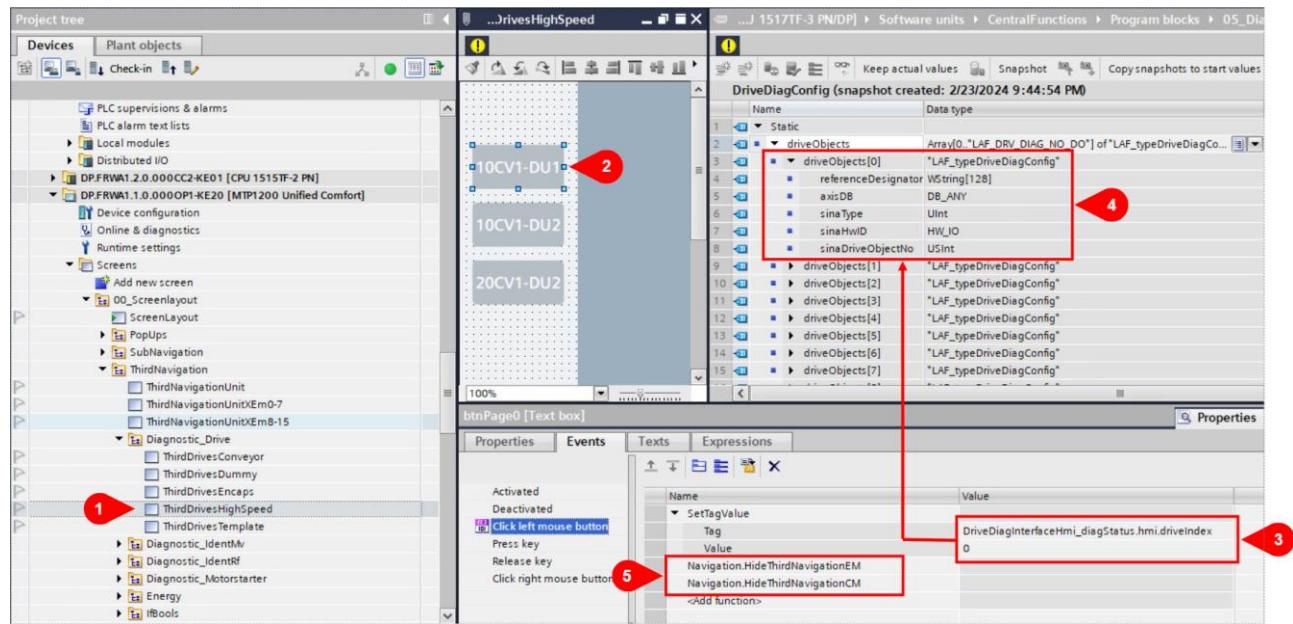


图 8-44 第三导航中的驱动器诊断事件

注 “DriveObjectIndex”的值必须与应诊断的“DriveDiagConfig”数据块中数组元素的下标匹配。使用相关的数组元素仔细检查该值。

最后导航到“属性”(1)，并将背景颜色动态化(2)的条件设置为与“DriveObjectIndex”变量(3)的事件中的相同值。

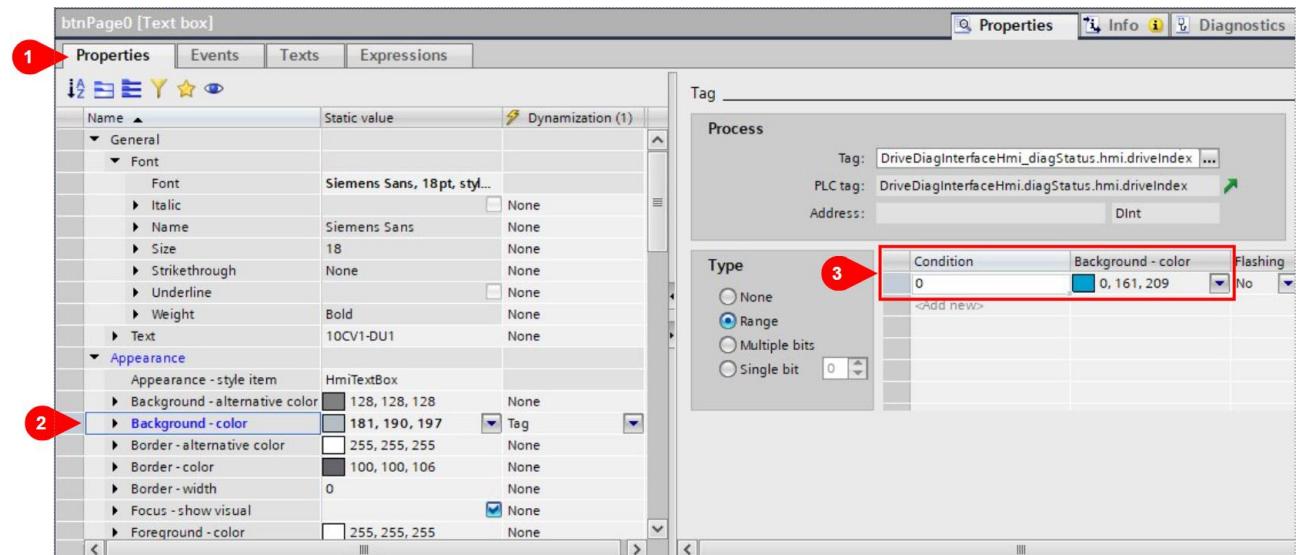


图 8-45 用于第三级导航中的驱动器诊断的动态化

8.7. 状态机模型

在人机界面中，该部分包含可视化状态模型和模式状态历史的屏幕。更多信息请参阅有关单元的章节以及 LUC（“单元控制库”）的文档。

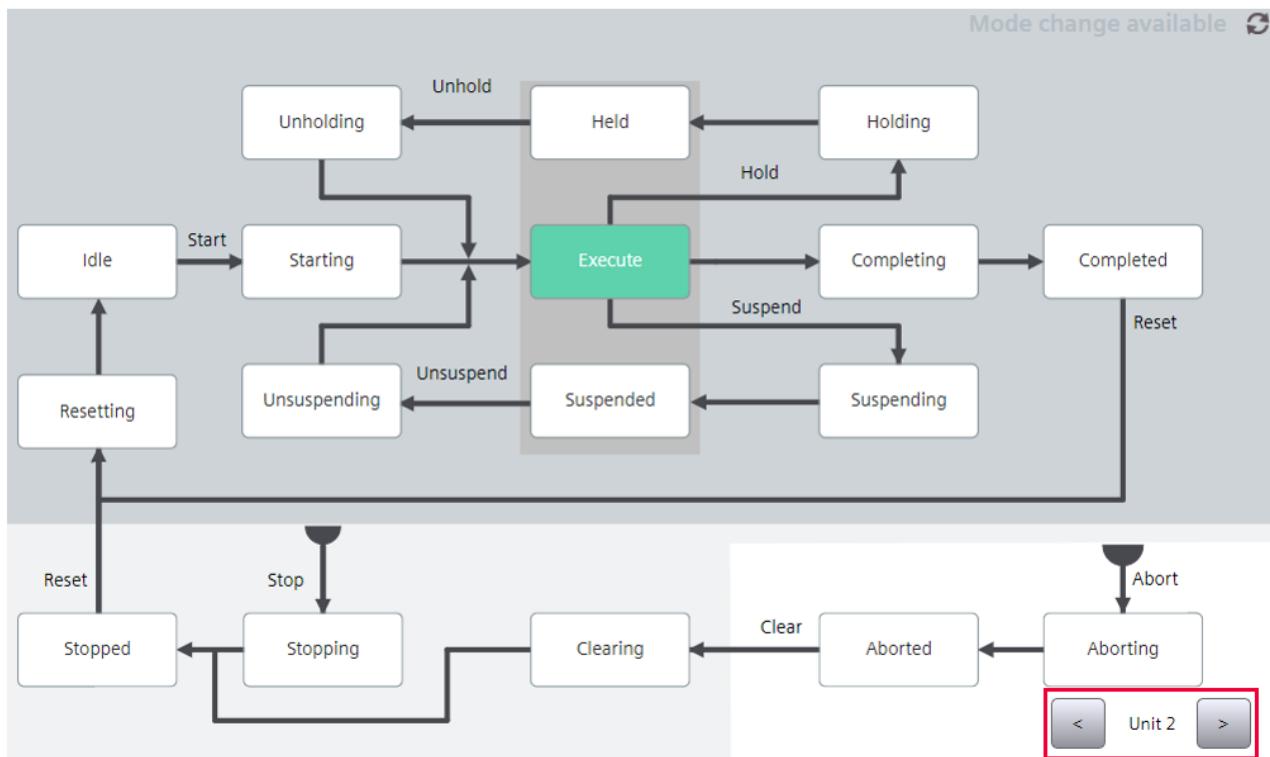


图 8-46: 状态模型界面

在屏幕右下角，可以更改显示的单元。

8.8. 网络服务器

例如，可以显示定义的网页：S7-1500 PLC 的嵌入式 Web 服务器（如果启用）。

授权用户可以通过网络上的 Web 服务器监视和管理 CPU。允许远程进行评估、诊断和修改。无需 STEP 7 即可完成监视和评估，只需打开 Web 浏览器。采取必要的预防措施来保护 CPU 免受损害非常重要，例如限制网络访问和使用防火墙。

也可以将“用户自定义页面”插入到网络服务器中。要创建自定义用户页面，可以使用 TIA Portal WinCC Unified V19 版本及以上的版本或任何 HTML 编辑器。

可以通过组态的 IP 地址访问 Web 服务器。在此示例中，可以通过以下地址访问 Web 服务器：<https://192.168.0.10>。

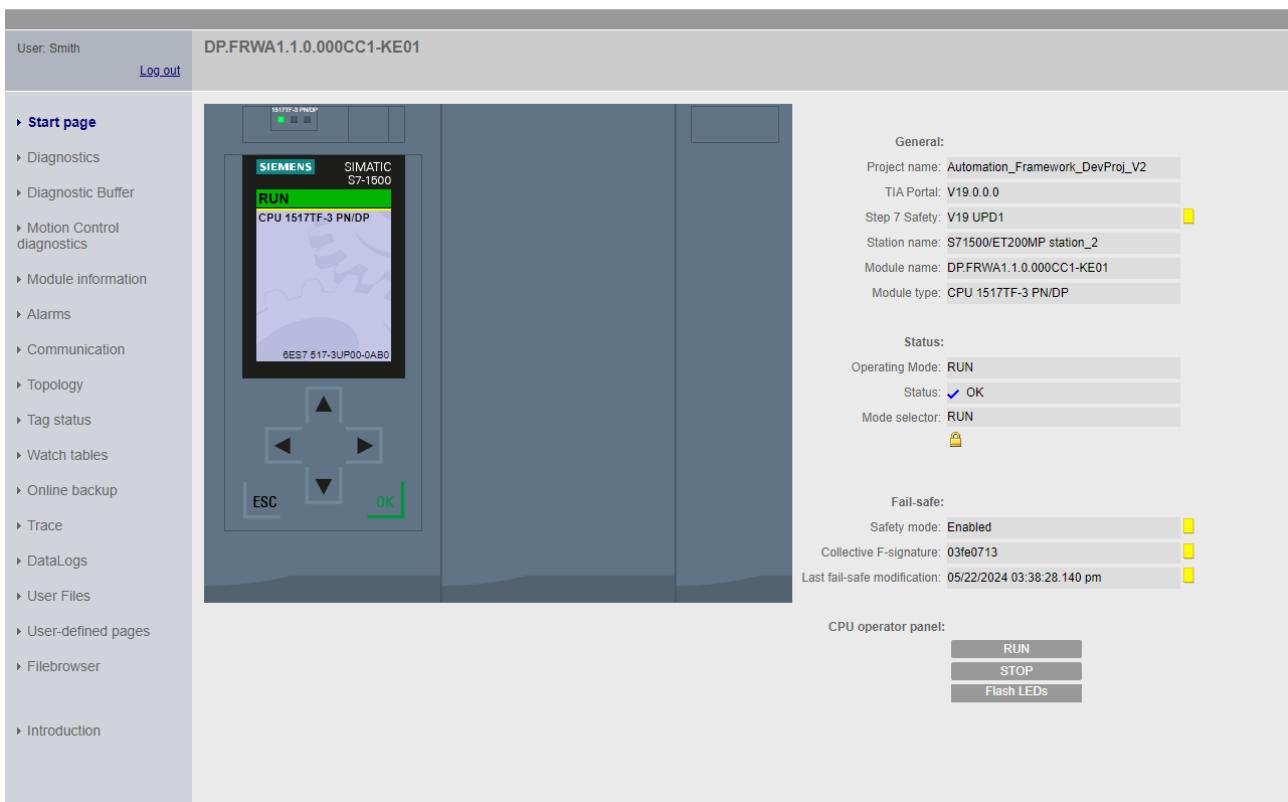


图 8-47 PLC S7-1500 的 Web 服务器视图示例

可以根据 UMAC 设置单独分配 Web 服务器的权限。可以在安全设置中进行此设置。可以在“角色”选项卡中设置权限。有关更多信息，请参见 Web 服务器的文档。

注

请注意，只有在 TIA Portal V19 中，才能在安全设置中对 Web 服务器的权限进行详细设置。如果使用旧版本，请查看 Web 服务器的文档 - 第 3.5 章“用户管理”。



有关 Web 服务器的更多信息，请参见以下文档：

<https://support.industry.siemens.com/cs/www/en/view/59193560>

8.9. 书签

此功能允许用户保存最多两个画面，以便直接从主导航跳转。

可以通过单击一个空星号来保存当前画面。完成此操作后，画面名称将出现在其旁边，星号颜色将设置为黄色。如下图(1)所示。

要删除书签，请单击创建书签时出现的叉号(2)。

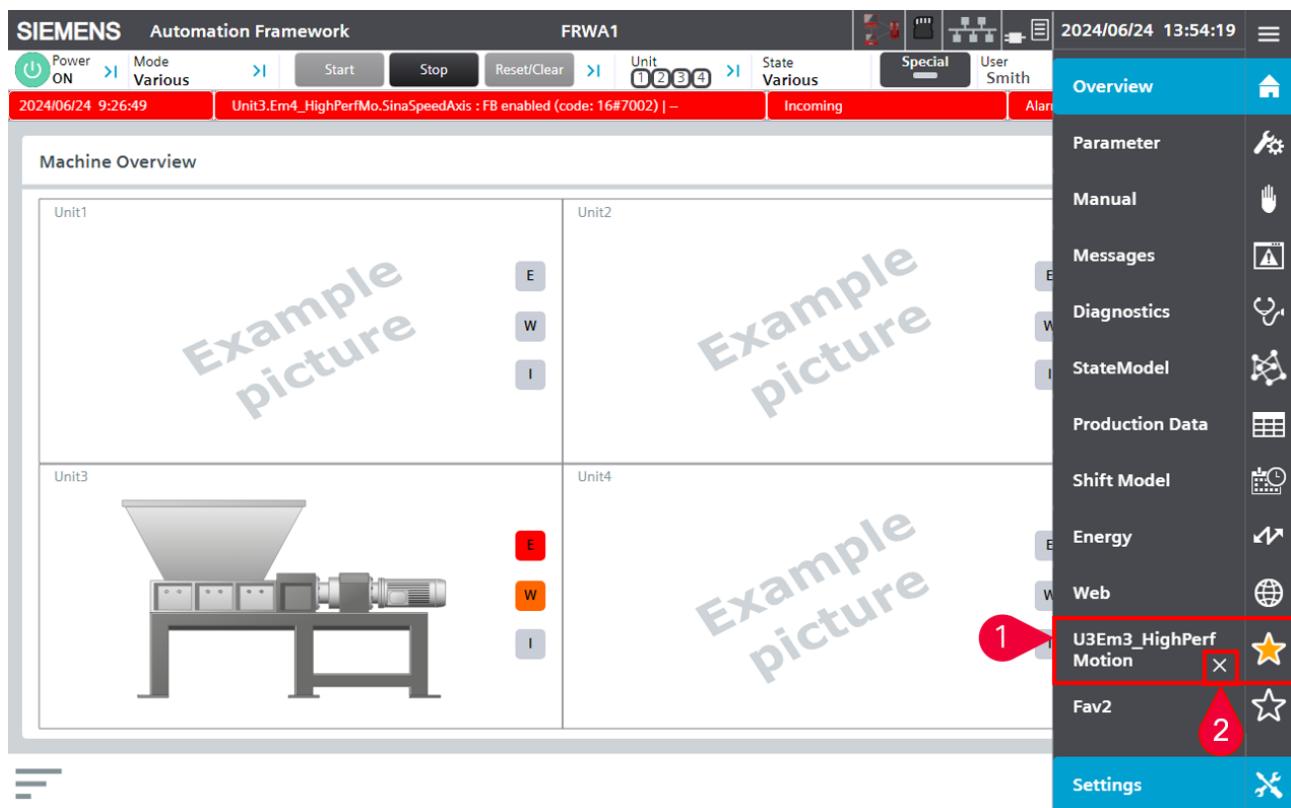


图 8-48 书签

8.10. 设置

设置涵盖常规 PLC 信息以及特定的 TIA Portal 项目信息。以下信息可在四个专用画面中找到：

- PLC & HMI
- PLC IM (识别和维护)
- 校验和
- 显示和更改当前日期和时间

PLC & HMI

PLC 和 HMI 画面提供有关 PLC 和安全循环时间 (1, 2) 的信息。显示当前以及最小和最大循环时间。这些参数的新读数可以通过按钮复位 (PLC/安全) 定时器触发。

还可以显示 PLC 中的预定义参数以及设定的通信负载。

此外，还介绍了对 SIMATIC 存储卡状态的评估。(3)



在 TIA Portal 项目中的“LAF_MemCardDiag”块上，通过按下 Shift + F1 组合键，可以找到有关存储卡诊断的更多信息。

此外，还实现了以下 WinCC Unified 面板功能：(4)

- 停止 WinCC Unified Runtime
- 打开控制面板
- 更新触发器
- 切换语言

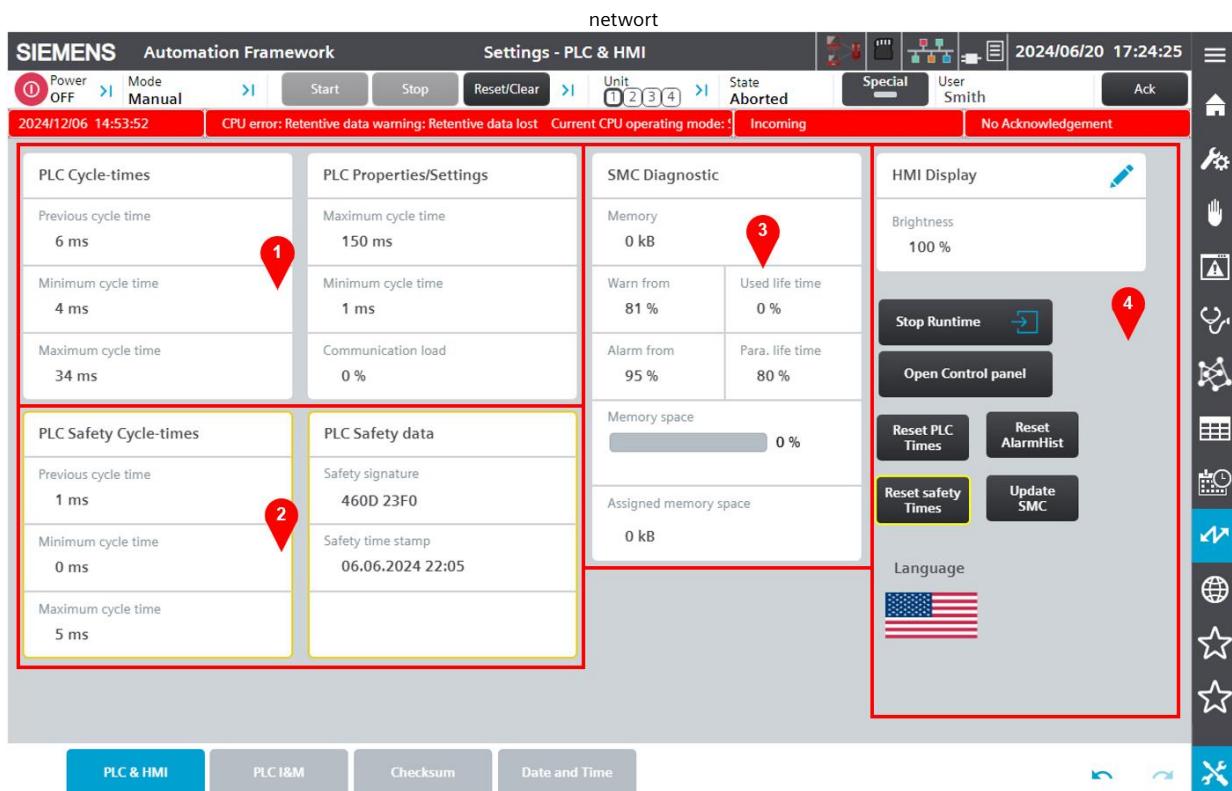


图 8-49 设置 PLC 和 HMI

PLC I&M

在 CPU 热启动后，将读取一次识别和维护数据以及 CPU 网络地址，并显示在下面的画面中。

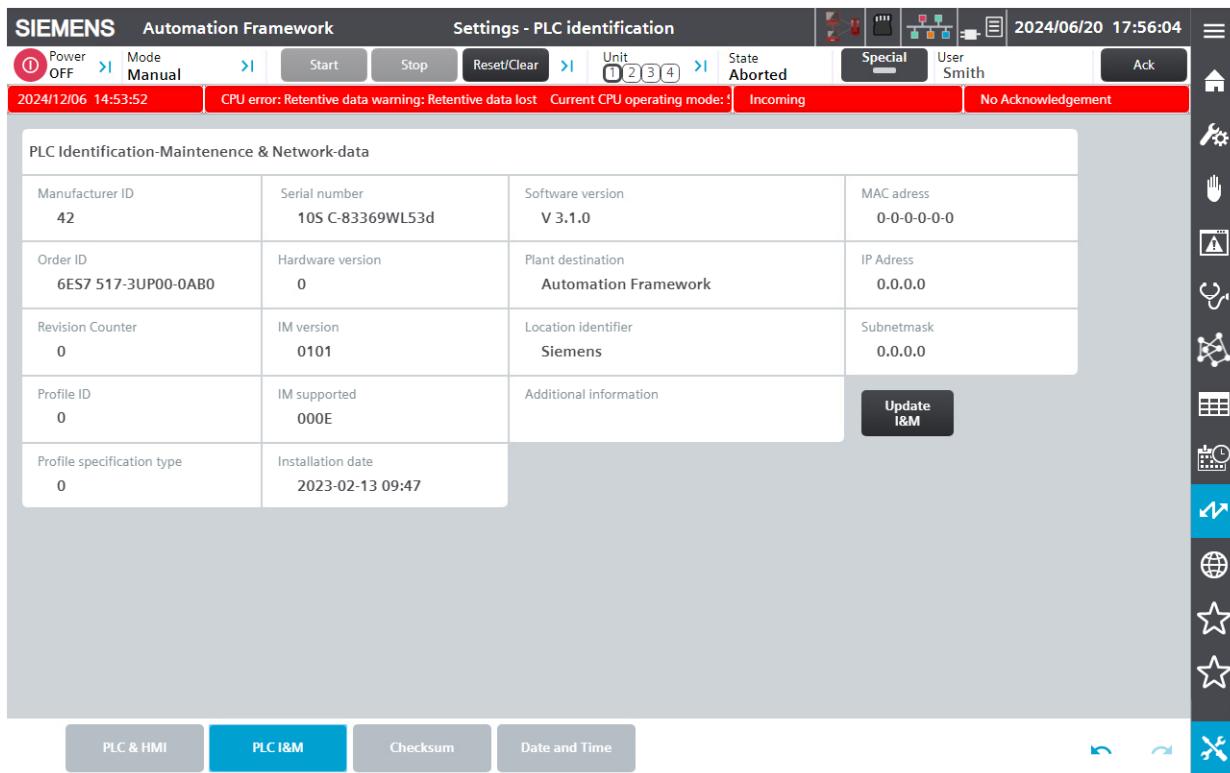


图 8-50 设置 PLC 和 HMI



在 TIA Portal 项目中的“LAF_PlcDiag”块上，通过按下 Shift + F1 组合键，可以找到有关 PLC 诊断的更多信息。

校验和

该块读取 PLC 标准块 (1)、PLC 安全块 (2) 和 PLC 文本列表 (3) 的校验和。将当前校验和与以前的值/旧值进行比较。对于每个偏差，都会发出 ProDiag 警告，并且画面“设置 - 校验和”中 WinCC 对象的颜色将以动画形式显示。

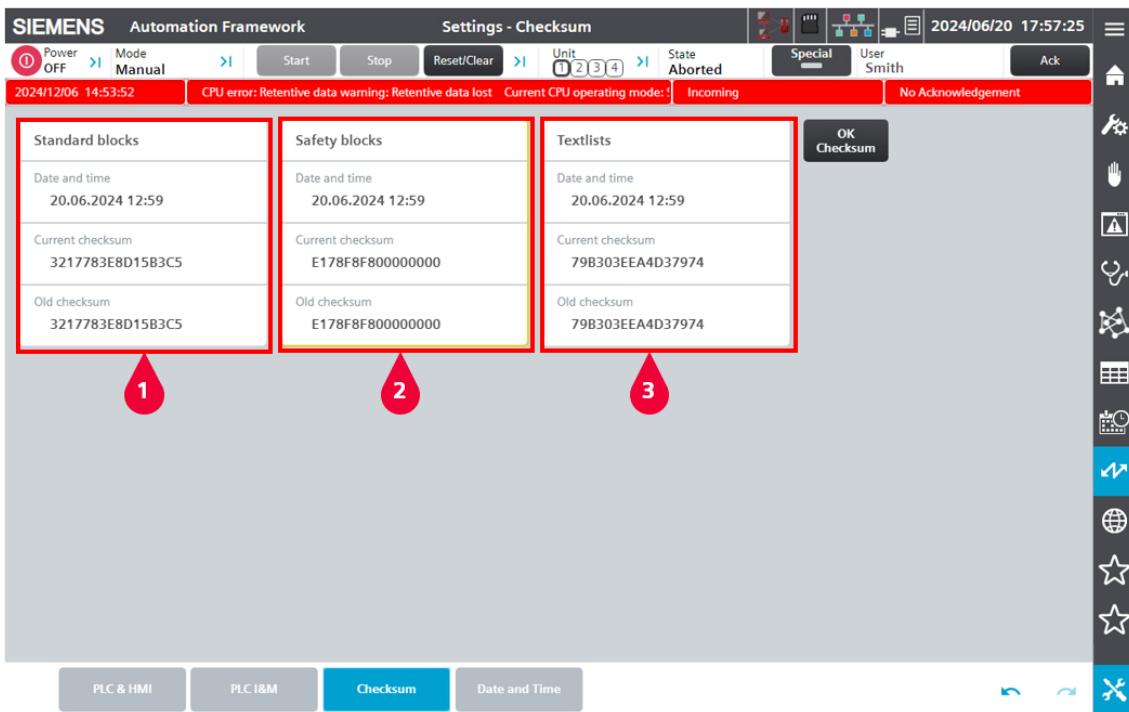


图 8-51 设置校验和



在 TIA Portal 项目中的“LAF_Checksum”块上，通过按下 Shift + F1 组合键，可以找到有关校验和的更多信息。

日期和时间

PLC 时间可以通过画面日期和时间进行编辑。期和时间信息将传递给 CentralFunctions → Programm blocks → 14_Settings → CallSettings。

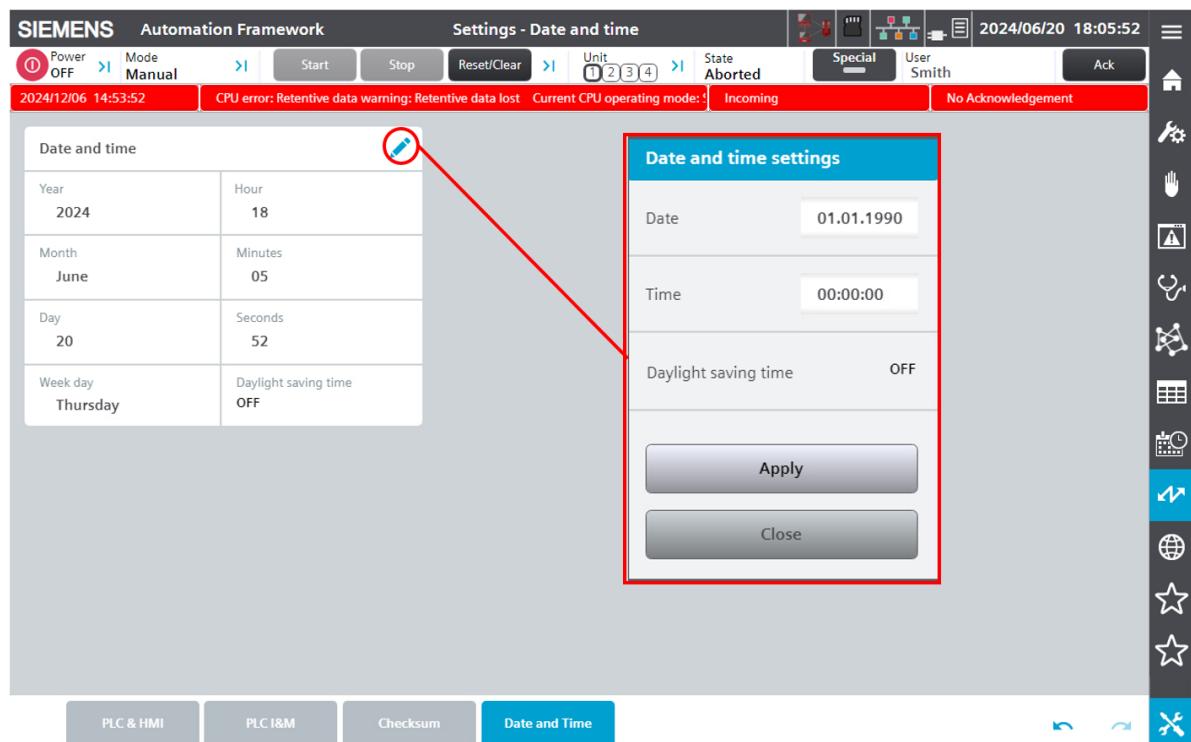


图 8-52 日期和时间设置

9. 单元

本章重点介绍这些单元。它描述了常规结构、接口、命令和数据流。

注

注意区分 ISA-88 单元和 TIA Portal 中的软件单元。本文档中的单元始终指 ISA-88 单元。软件单元明确表示为软件单元。

AF 项目中实现了四个单元：

- 单元 1：使用 SCL 编程的示例实现单元。
- 单元 2：使用 LAD 编程的示例实现单元。
- 单元 3：演示实现单元，涉及演示 EM 的处理。在 LAD 中编程。
- 单元 4：可用作启动的虚拟单元，使用基础 EM 设置自定义单元。

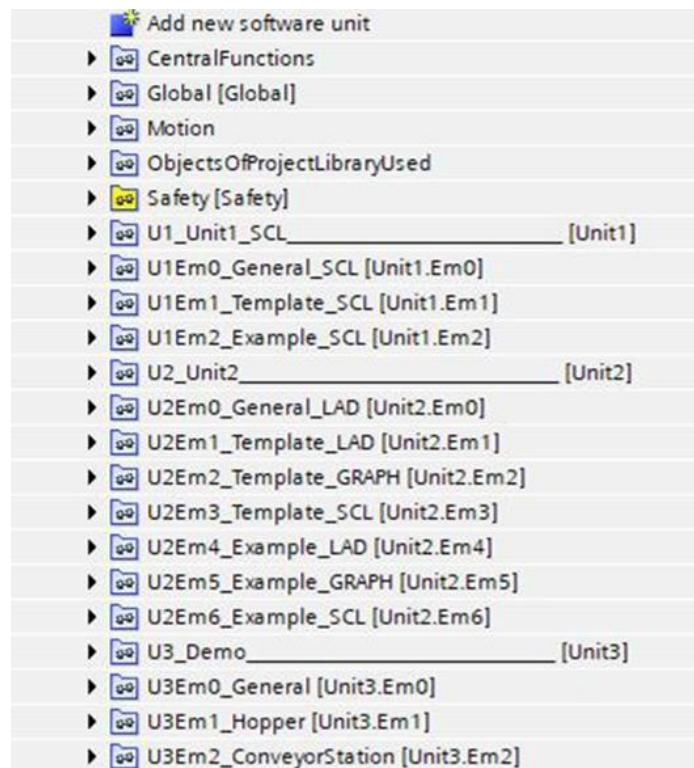


图 9-1 该单元和 EM 的软件单元

9.1. 常规结构

每个单元由多个软件单元组成。包含表示单元本身的第一个软件单元和代表设备的每个子 EM 的附加软件单元。

主程序结构

- 单元级别正在根据单元被分配到的安全区域处理安全评估和响应
- 处理和管理内部告警
- 处理状态和模式管理器以及状态和模式管理器本身的反馈、接口和后处理，这将在下一章中进一步讨论。

信号流

信号流首先评估来自 EM 的当前反馈和从自身模块生成的反馈以及安全状态。该评估是使用 FB “LUC_StatusCollector” 进行的。旨在创建当前过程接口。然后，该接口与外部接口一起进行评估，例如来自 HMI 的数据，即操作员执行的命令，或外部信号，例如来自 FB “LUC_InterfaceManager” 中的 IT 系统（例如 OPC UA）的信号。该块由此生成一个适配的接口，该接口适用于 LPML 块，并请求模式和状态更改或传输 StateComplete 消息。然后，UnitModeStateManager 评估命令并设置当前模式和状态，这些模式和状态通过 “LUC_SetUnitStatus” 函数传输到 DB “EmNodes”。由此，EM 依次读取当前单元状态并执行其逻辑，并在此循环中设置新的反馈信号，然后在新循环中再次评估这些反馈信号。

一个单元的整体信号流如下图所示。

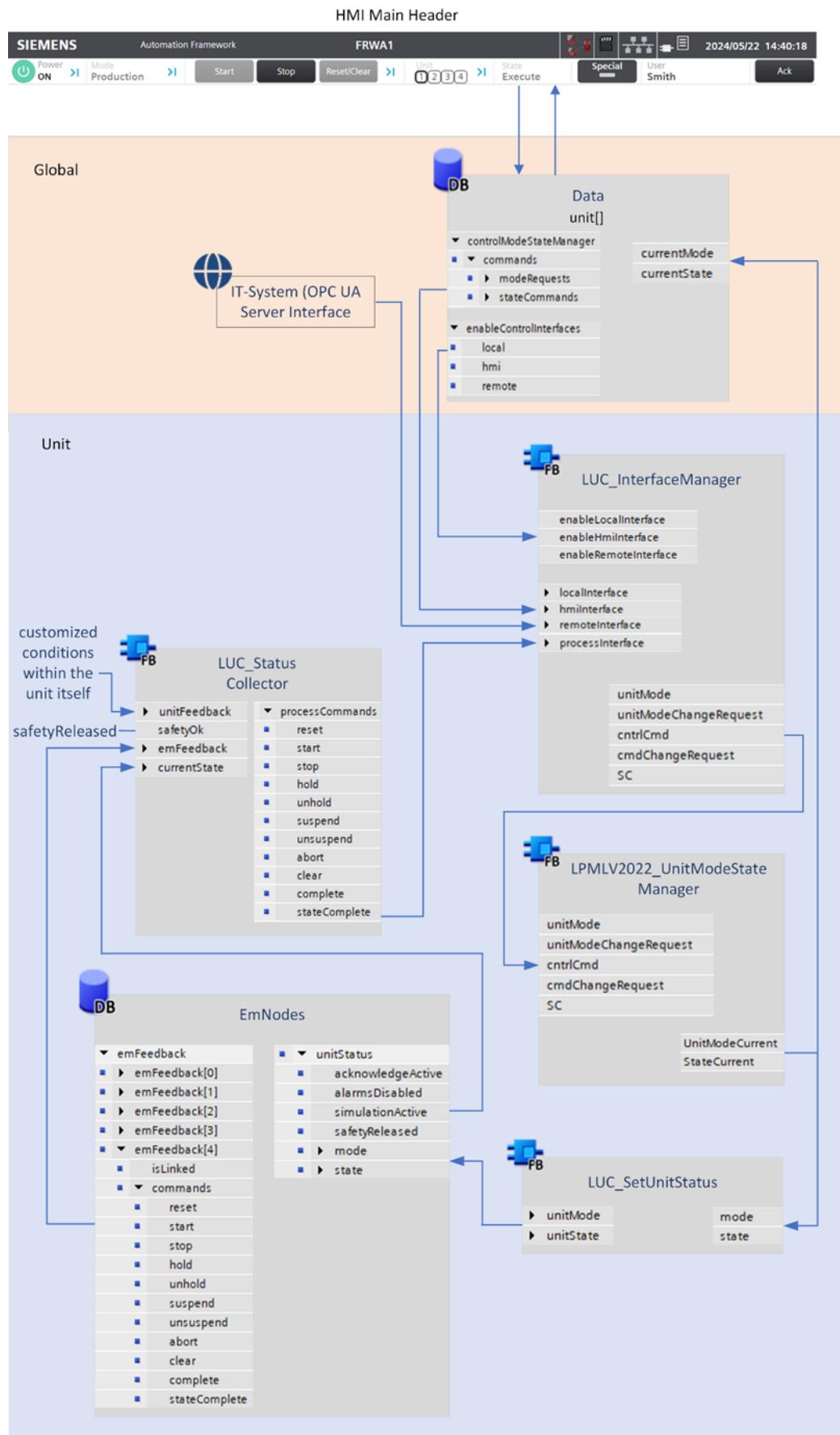


图 9-2 单元的信号流

9.2. 接口和块

单元的常规程序架构基于[程序结构](#)一章中的描述。为了具体说明级别和所用对象之间的差异，以下章节更详细地描述了单元级别和在 AF 中实现的单元的不同方法。

关系

该单元需要将关系组态为以下软件单元，如图所示。

Relations		
Selected unit	Relation type	Accessible element
U3_Demo		
U3_Demo	Software unit	ObjectsOfProjectLibraryUsed
U3_Demo	Software unit	Global
U3_Demo	Software unit	Safety

图 9-3 单元程序块的关系

一个单元包含以下块：

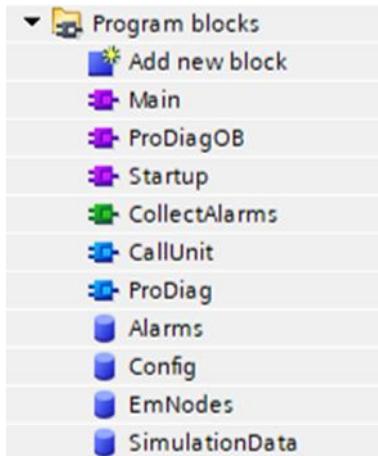


图 9-4 该单元的程序块

LPM LV2022（PackML 库）中的块主要用于模式状态管理器，LUC（单元控制库）用于单元内的补充信号处理，LAF（自动化框架库）用于处理相应单元内的报警管理。为了准确说明特殊块如何在单元级别运行，除了已经讨论过的基本功能外，下面还将讨论单元内的具体实现。

OB Startup（单元）

状态-模型的组态：

用户可以根据自己的要求调整状态机。这是可选步骤，仅当用户具有不同的状态机（如标准 OMAC PackML 状态机）时才需要执行此操作。

在 OB “Startup” 中，FB LUC_StateModelConfig 将用户设置传输到状态模式管理器组态中，状态模式管理器在运行期间使用该组态。

DB Config（单元）

在单元 DB “Config” 中，用户可以调整以下组态设置：

- 相对级别和父级别的引用指示符
- 信号堆栈行为
- 组态哪些 OPC UA 方法可以远程控制状态机
- 基于 OMAC PackML 状态机的状态模型组态

DB EmNodes (单元)

该 DB 包含单元的状态、设备模块的反馈、单元的不同 EM 之间的关系信号以及 signalStack 的当前状态。

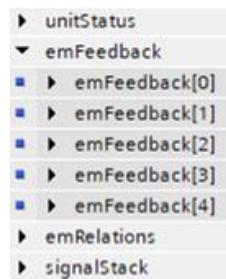


图 9-5 单元级别的 EmNodes 数据块

在 AF (U3_Demo) 的演示单元中，该机器有五个设备模块，这些模块被分配给单元级别。Em4 和 Em5 仅用于展示如何实现不同的方法，但在单元到 EM 行为方面不起作用。

每个设备模块都将其当前状态写入 emFeedback 的相关数组元素中。此状态在单元逻辑中评估，并影响状态机。同样，该单元将其当前状态写入 unitStatus 中。这种状态在每个设备模块的逻辑中进行评估，它们根据当前模式和状态运行其作业/序列。emRelations 信号可用于在不同的 EM 之间传输信息。这种信号交换集中在设备的“EmNodes”数据块中，以查看以下级别当前正在处理的内容。signalStack 的数组可用于连接 Em0 内的 Stack Light 等硬件对象。

DB SimulationData

此全局 DB 保存仿真 TIA Portal 项目的值，无需硬件设置。

OB ProDiagOB / FC CollectAlarms / FB ProDiag

这些块用于自动化框架中所实施的警报概念。有关如何使用这些块的进一步说明，请参见“[诊断原理](#)”一章。

FB CallUnit

FB CallUnit 块执行以下操作：

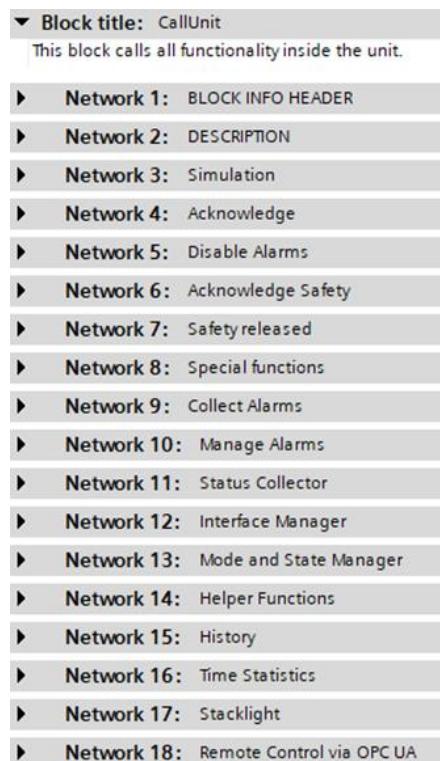


图 9-6 FB “CallUnit”的内容

仿真/确认/禁用报警

如果仿真、确认或禁用报警在全局级别上处于活动状态，则继承此状态也会在此单元中设置为活动状态。信息被写入数据块“EmNode”，从那里分发到设备模块。

确认安全/安全释放

需要安全确认时，如果状态机处于相应状态或在全局级别设置了确认，则该单元将向安全程序发送确认命令。该单元评估指定安全区域的安全释放，并做出相应的反应。如果发生安全事件，它会向状态模式管理器发送中止命令。

特殊功能

该单元的特殊功能命令被复制到数据块“EmNodes”中，以便所有 EM 都可以相应地继承它们并做出反应。

收集报警

在收集报警功能中，可以将任何事件分配给报警。ProDiag 可以监控此特定报警，并将其映射到设备的响应（例如中止、停止、保持）。有关 AF 中过程诊断的更多信息，请参见“[诊断原理](#)”一章。

管理报警

此块从 ProDiag 监控派生命令。ProDiag 监控包含定义这些监控响应的类别（例如，中止、停止、保持）。它们将作为命令移交给状态采集器。

状态采集器

在这里，反馈是从单元级别（命令）和所有底层设备模块（命令、stateComplete）收集并合并的。块将一组汇总的命令移交给接口管理器。

接口管理器

评估并总结了 ModeState 管理器的不同命令源。这可能是操作员通过 HMI 或 IT 发出的命令，也可以是由于过程状态评估而产生的状态更改请求。

模式和状态管理器

此块表示机器的模式/状态模型。它始终输出机器的当前模式和状态。它评估模式更改请求、状态更改请求或 stateComplete 输入是否处于挂起状态，并相应地更新模式和状态（如果允许）。

Helper Functions

模式状态管理器的进一步信号处理，以与单元级的数据结构进行交互。

历史/时间统计

此块更新模式和状态的历史记录。当模式或状态发生更改时，当前模式和状态将按升序存储在历史记录的当前数据区域和以前的数据区域中。然后，时间统计程序段内的 UnitModeStateTimes 块评估单元前一种模式和状态的运行时间。

堆叠灯

该程序段根据组态以及当前模式和状态设置信号堆栈输出。

通过 OPC UA 进行远程控制

该程序段中的“LUC_RemoteCtrl_OpcUa”块用于通过 OPC UA 方法调用来控制状态机。

9.3. HMI 画面

要访问单元的概览画面，操作员可以从整个机器的主概览中选择该单元 (1)，或者通过导航栏进入此处，如果按下左上角的图标 (2)，导航栏将打开。出于模拟目的，模拟激活按钮被添加到主概览画面上。此外，安全区的集体信号通过虚拟的 E-Stop 按钮显示。如果所有安全区均已释放，则该按钮将被停用。但是，如果任何区域未释放，则该按钮将显示为已激活。



图 9-7 整体机器视图

如果做出选择，则设备的概览画面将打开并显示不同的设备模块。单元 3 的演示实现如图 9-8 所示。对于客户项目，可以在此处添加具体的机器原理图或画面。出于仿真目的，虚拟急停按钮和仿真激活按钮也被添加到主概览画面上。操作员可以通过单击概览画面中的设备来访问各个设备画面，也可以使用导航栏，该导航栏还包括机器的 EM 级别。设备模块画面的结构将在“HMI 画面”一章中介绍。

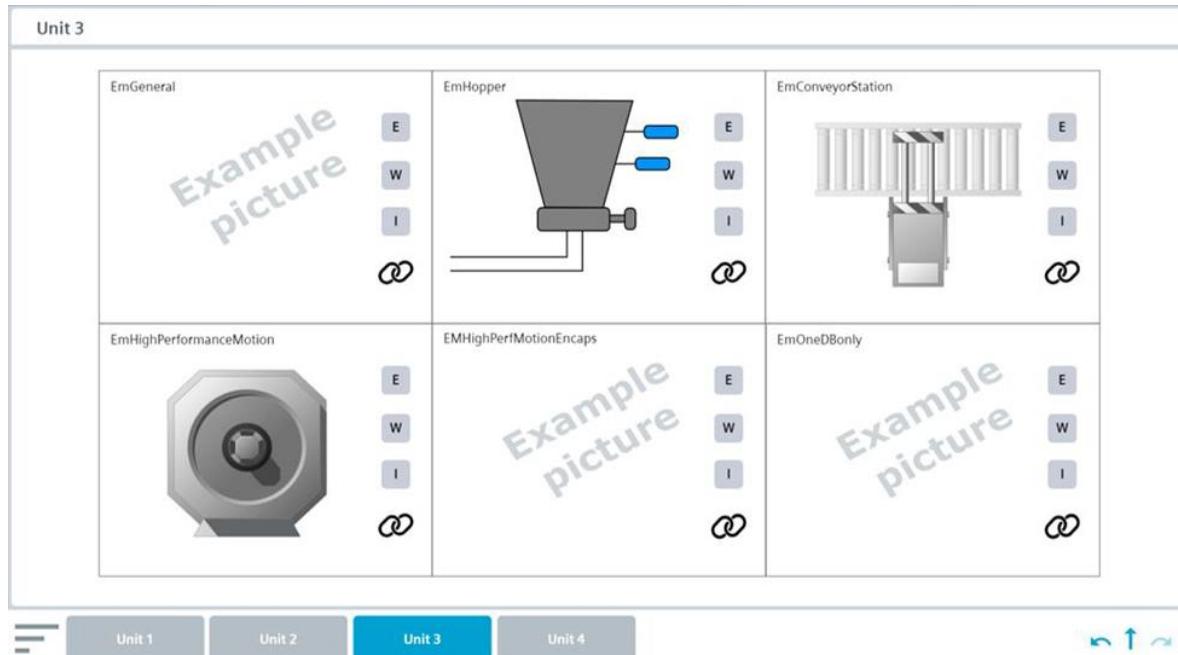


图 9-8 演示单元 3 概览画面

9.4. 模式和状态管理器

AF 使用 [LPML](#) 库，该库提供符合 OMAC PackML V2022 标准的模式和状态管理器（包括补充块）的实现。此外，单元控制库（[LUC](#)）还用于进一步相关的模式操作任务，例如命令和状态收集以及不同命令源的接口管理。

模式和状态管理器在每个单元的 FB “CallUnit” 中调用。它是通过 [LPML](#) 库的 FB “LPMLV2022_UnitModeStateManager” 来实现的。有关该块的详细信息，请参见该库的文档。该块的主要功能是评估外部接口所请求的模式和状态，是否可以进行模式或状态更改并将当前模式和状态提供给其他程序部分。

组态

自动化框架提供了一种简单的方法来实现状态机所需的组态，方法是将另一种基于 PLC 数据类型

“LUC_typeStateMachineConfiguration”的数据结构添加到单元的“Config”数据块中。因此，此数据块内部有两个数据结构。一个用于用户的组态，另一个是“LPMLV2022_UnitModeStateManager”使用的 UDT。以下参数可根据自身需求进行调整：

- 使用的模式
- 每种模式的使用状态
- 模式之间允许的过渡
- 被允许过渡到保持或完成过程的状态

用户可以在 enableModes、enableStates、enableModeTransitions、enableHoldCommand 和 enableCompleteCommand 中定义状态模型组态。每种模式都有自己的状态和过渡组态。保持和完成命令的组态独立于当前模式，并定义了哪种状态可以转换为保持或完成状态。

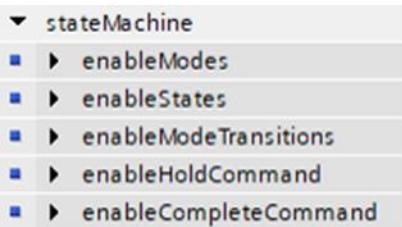


图 9-9 在单元的 DB “Config”中组态状态机

要启用新模式或禁用旧模式，用户可以在 enableModes 变量中设置相应的位。下一步是定义不同模式中存在的状态。为此，enableStates 数据结构将每个状态作为每个模式的布尔变量包含在内。enableModeTransitions 定义在哪种状态下可以更改模式。此数据的结构相同，因此每种模式都具有每种状态的过渡激活。此外，通过设置或不设置位，可以组态是否可以从某个状态转换到“保持”或“完成”状态。

HMI 接口

HMI 的模式和状态管理器的数据集中存储在 Software Unit Global 的数据块 “Data” 中。每个被规划的单元都包含了如下图所示的数据结构。该接口使状态机可以通过 HMI 进行控制，并读取当前状态。

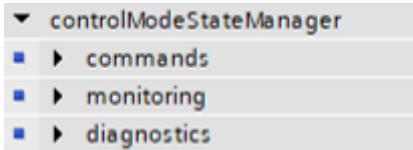


图 9-10 位于软件单元 Global 的 DB “Data”中的 ModeStateManager 接口

历史和统计

每个单元在数据块 “Data” 中的 UDT “history” 中还存储了其之前的状态和模式的历史记录。提供了状态机的最后 15 个步骤。FB “LUC_UpdateModeStateHistory”（在 FB “CallUnit” 中调用）会在每次模式或状态发生变化时更新该历史记录。

所有模式和状态的时间统计信息都存储在 UDT “modeStateTimes” 中。这提供了状态机在各个模式和状态中停留的时间的详细概述。FB “LPMLV2022_UnitModeStateTimes”（在 FB CallUnit 中调用）会不断更新该统计信息。

画面

从 HMI 使用主标题上的模式和状态命令按钮控制模式和状态（参见图 9-11）。此外，从 HMI 右侧的主导航中，可以打开特定状态模型画面。此画面提供所选模式的已组态状态和当前状态的概览。另一个画面显示状态机的历史记录（参见图 9-12）。



图 9-11 状态命令和模式命令弹出窗口

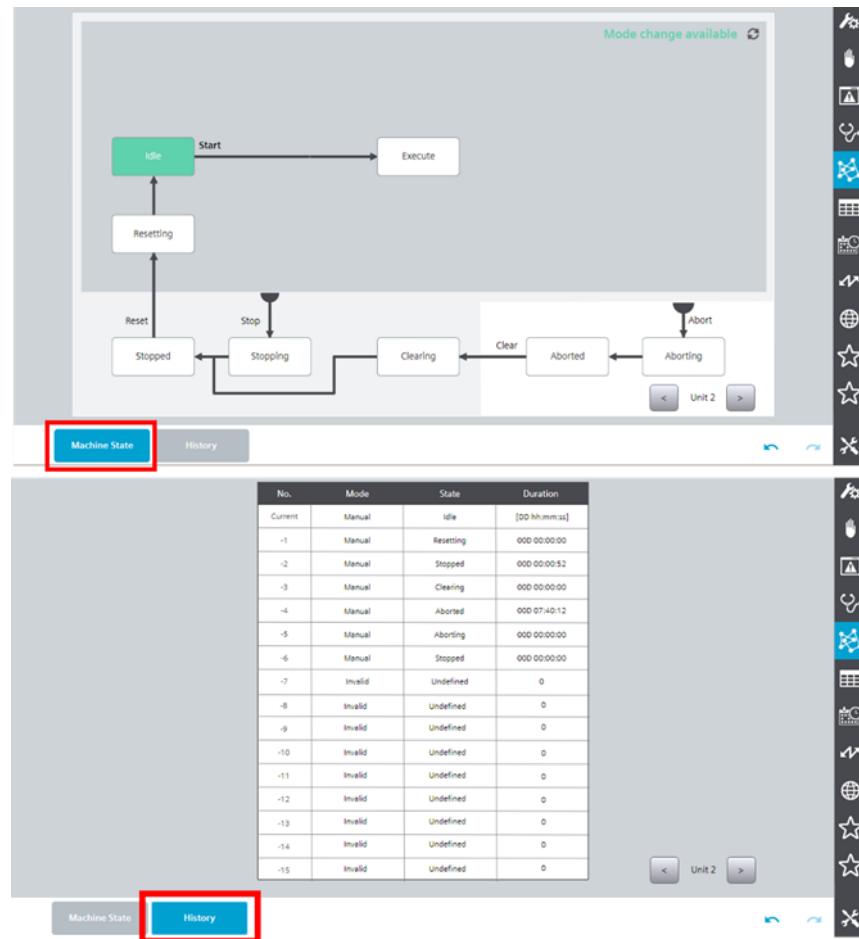


图 9-12 状态模型视图和历史记录

注

对西门子应用示例 [\[5\]](#) 的 OMAC 状态模型的可视化进行了调整，以符合 AF 西门子风格指南

9.5. 堆叠灯

每台机器通常至少有一个堆叠灯，用于向设备操作员指示机器状态。AF 中的堆叠灯支持五种标准堆叠灯颜色，具有各种照明选项（静态或闪烁），并为额外的用户自定义颜色以及声音指示器的开/关信号提供了足够的储备。

下图显示了映射到单元不同状态的堆叠灯的实现，在 AF 中实现：

	Undefined	Clearing	Stopped	Starting	Idle	Suspended	Execute	Stopping	Aborting	Aborted	Holding	Held	Unholding	Suspending	Unsuspending	Resetting	Completing	Completed
Red Lamp Flashing	F F								F F									
Red Lamp Solid			S					S								S	S	
Yellow Lamp Flashing														F				
Yellow Lamp Solid						S												
Blue Lamp Flashing										F		S						
Blue Lamp Solid																		
Green Lamp Flashing						F												
Green Lamp Solid	S					S							S	S	S			
Horn Flashing																		
Horn Solid																		
White Lamp Flashing																		
White Lamp Solid																		
User1 Lamp Flashing																		
User1 Lamp Solid																		
User2 Lamp Flashing																		
User2 Lamp Solid																		

图 9-13 PackML 状态模型映射到 stacklight

AF 提供了此类单元堆叠灯的预组态实现。该示例是根据 IEC 60073 实施的，符合 OMAC 准则第 6 部分：PackML 用户界面 – HMI。堆叠灯在单元的“Config”DB 中组态，如图 9-14 所示。

注

请记住，堆叠灯功能尚未在 AF 项目中完全实现。

Name	Data type
1 Static	
2 stateConfiguration	Array[0:_LPMLV2022_MAX_MODES_UPPER_LIM] of _KE01.LibraryObjects.LUC_typeManagerStateConfiguration
3 transitionConfiguration	Array[0:_LPMLV2022_MAX_MODES_UPPER_LIM] of _KE01.LibraryObjects.LUC_typeStates
4 holdConfiguration	_KE01.LibraryObjects.LUC_typeStates
5 stateModeManager	_KE01.LibraryObjects.LPMLV2022_typeConfiguration
6 remoteCtrlOpcUa	_KE01.LibraryObjects.LUC_typeRemoteCtrlConfiguration
7 signalStack	_KE01.LibraryObjects.LUC_typeSignalStackConfiguration
8 referenceDesignator	String[20]
9 blinkFrequency	Real
10 hornPulseTime	Time
11 states	_KE01.LibraryObjects.LUC_typeSignalStackState
12 undefined	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
13 clearing	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
14 static	_KE01.LibraryObjects.LUC_typeSignalStackPeriphery
15 flashing	_KE01.LibraryObjects.LUC_typeSignalStackPeriphery
16 stopped	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
17 starting	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
18 idle	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
19 suspended	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
20 execute	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
21 stopping	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
22 aborting	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
23 aborted	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
24 holding	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
25 held	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
26 unholding	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
27 suspending	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
28 unsuspending	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
29 resetting	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
30 completing	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour
31 complete	_KE01.LibraryObjects.LUC_typeSignalStackStateBehaviour

图 9-14 AF “Config”数据块中的堆叠灯

9.6. 演示单元

在单元 3 中，演示实现了 EM 的处理。该单元的结构如下：

U3_Demo

在 LAD 中编程实施的单元，包括：

- 安全和电源释放处理
- 级别本身和底层模块内的报警处理
- 评估相关的 EM 状态
- 从 HMI 或通过 OPC UA 远程控制处理单元接口
- 状态模式管理器，包括 OMAC 模型

U3Em0_General

这是一个设备模块的演示实现，重点介绍：

- 整个单元所需的逻辑
- 包括常规发布和系统级功能，例如机器堆叠灯的处理

U3Em1_Hopper

这是一个设备模块的演示实现，演示了：

- 如何使用技术模块（多个控制模块的组合使用）
- 如何使用完全基于 LAD 编程的步骤序列实现不同的作业

U3Em2_ConveyorStation

这是一个设备模块的演示实现，演示了：

- 如何使用 S7-Graph 作为主要序列
- 如何使用 S7-Graph 序列实现不同的作业

U3Em3_HighPerformanceMotion

这是一个设备模块的演示实现，演示了：

- 如何在运动控制周期中调用高性能运动轴
- 高性能运动编程如何与设备模块序列相关联

U3Em4_HighPerformanceMotionEncaps

这是一个设备模块的演示实现，演示了：

- 如何使用相同的高性能运动 EM 封装（所有内部使用的信号都作为接口传递到设备模块调用）

U3Em5_ExampleOneDbOnly

这是一个设备模块的演示实现，演示了：

- 如何将设备模块的所有数据存储在一个 DB 内。因此，ControlNodes、Config 和 HmiInterface 值都分配给此 DB 中的 EM 对象

9.7. 如何添加新单元？

要添加新单元，可以使用现有单元复制并粘贴到软件单元结构中。例如，将添加另一个称为 U5_Unit5_NewUnit 的单元。

- 在“Software Unit Global”中，转到“PLC 变量”，然后选择全局 PLC 变量表。在“用户常量”选项卡中，必须执行以下步骤

Global				
	Name	Data type	Value	Comment
	UNIT05	Int	4	Unit5
	UNIT04	Int	3	Unit 4
	UNIT02	Int	1	Unit 2
	UNIT03	Int	2	Unit 3
	UNIT01	Int	0	Unit 1
	NO_OF_UNITS	Int	4	Highest value of unit (Counting starts from 0)

图 9-15 全局 PLC 变量列表

- 将 NO_OF_UNITS 增加 1，以显示整台机器中最后一个单元的最高值。
- 使用附加单元的名称添加一个新常量，并分配下一个可能的更高数字。
- 复制 AF 的现有单元，然后通过右键单击 PLC 项目树中的 Software Units 文件夹然后选择粘贴，将其粘贴到项目中。
- 在新的软件单元中，根据需要调整名称和命名空间。因此，右键单击新的软件单元并打开属性。
- 输入新的名称和命名空间后，单击“应用于所有底层元素”，将命名空间应用于软件单元内的所有对象。

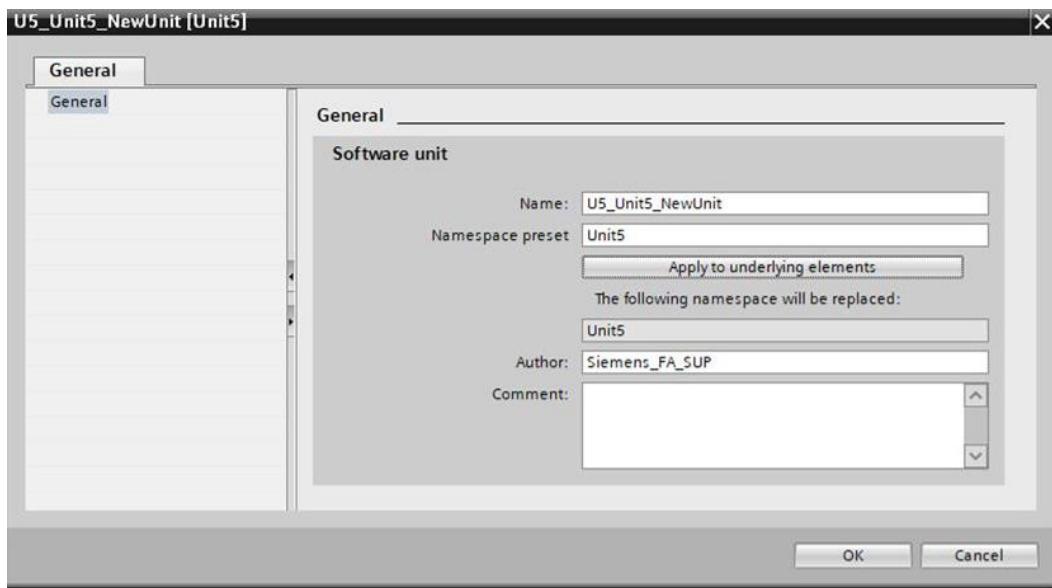


图 9-16 新软件单元的属性

- 打开新单元的 OB “Main”，并在调用 FB CallUnit 之前调用的 Move 块中分配在 Global Software Unit 中创建的用户常量
- 通过将相应的接口连接到 safetyReleased 输入和 acknowledgeSafety 输出，为设备设置正确的安全区域。

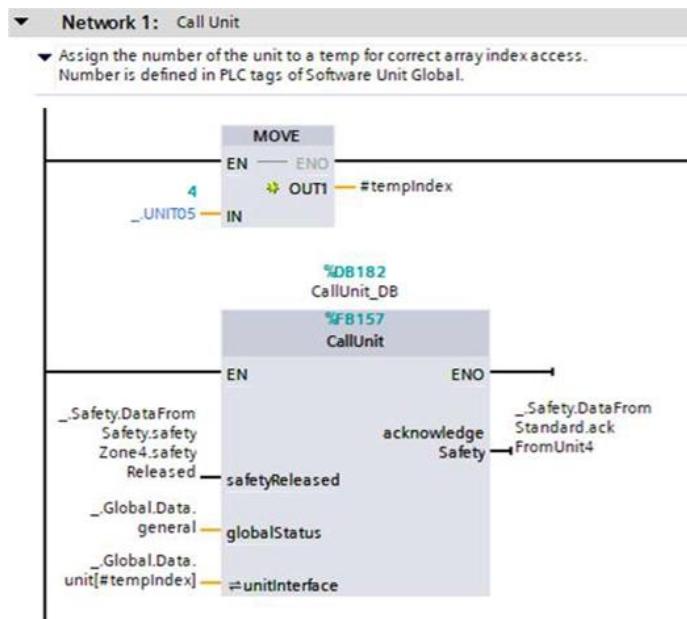


图 9-17 新单元的 OB “Main”

- 根据 [PLC 启动和循环调用序列](#)一章中介绍的 OB 编号规则更改 OB “Main”、“Startup” 和 “ProDiag”的 OB 编号。

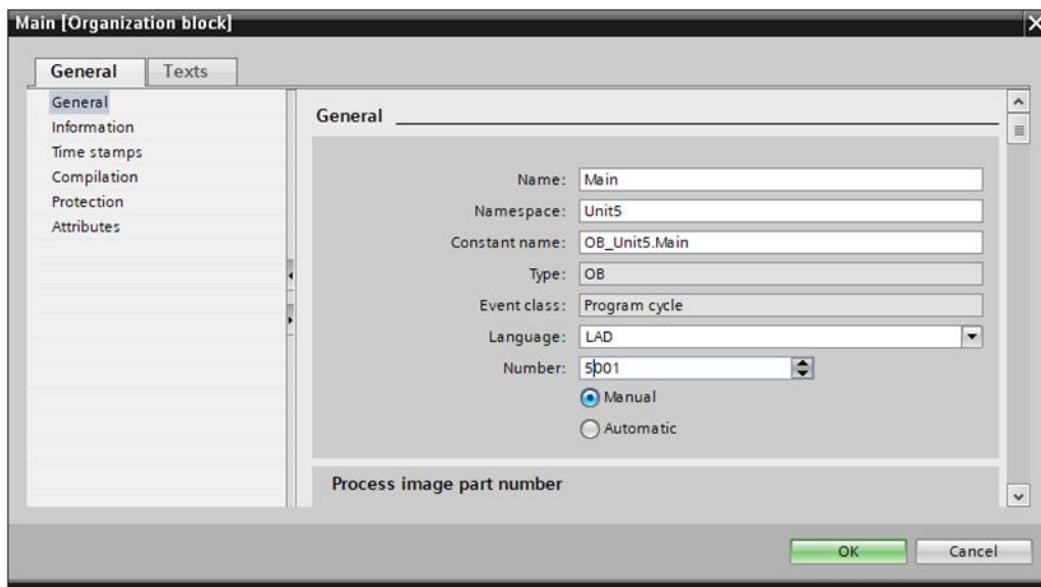


图 9-18 OB “Main”的属性

- 通过编译 PLC 对其余块重新编号，并选择自动解决冲突。
- 然后可以根据需要组态该单元。

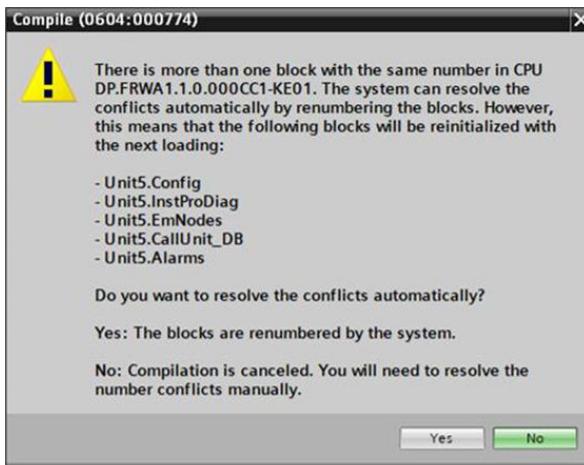


图 9-19 提示解决块编号问题

9.8. 如何移除 HMI 上的单元

在此示例中，将删除 Unit4。因此，应调整并删除所涉及到的多个画面和脚本，以便能够减少单元数量。

首先，应标识该单元的文件夹：

- 位置：在屏幕层次结构中确定要删除的 Unit。观察它与其他单元和画面的关系。

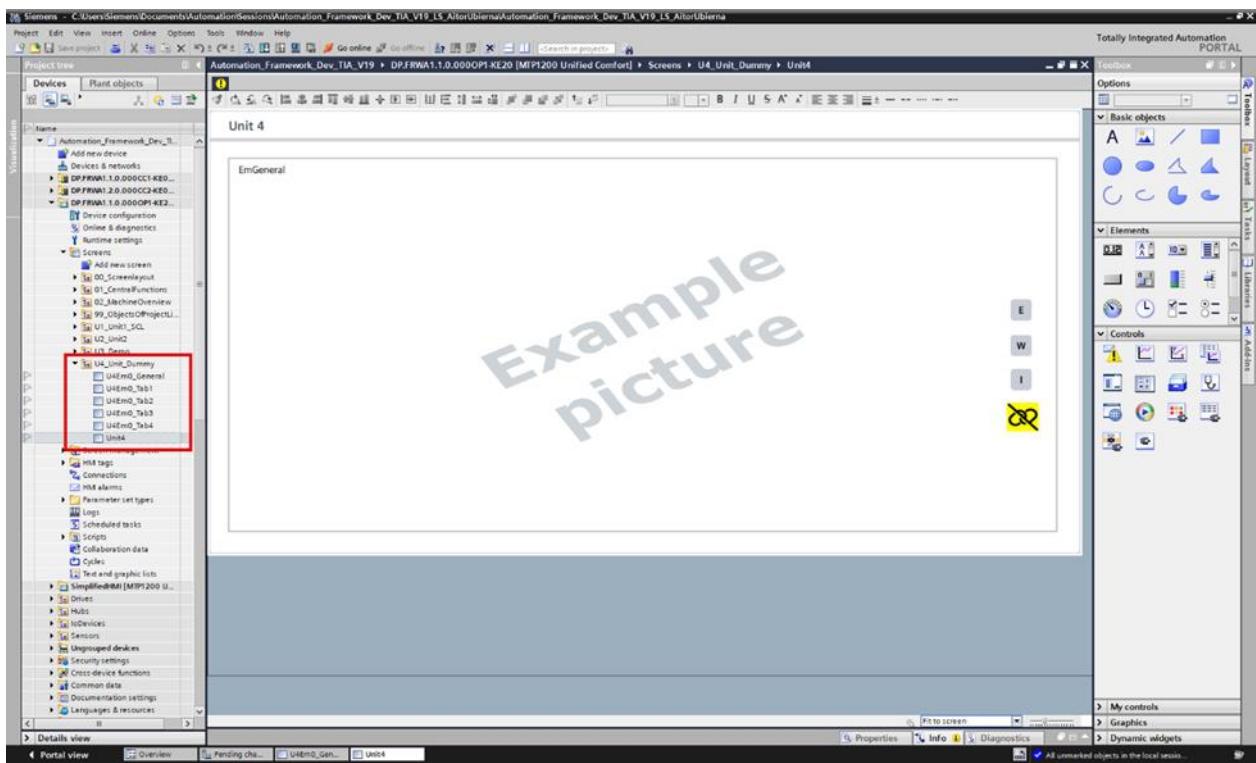


图 9-20 标识所需单元的画面文件夹和层次结构。

此外，应调整 Machine Overview 画面以删除已删除单元的组。

其它画面上的元素：

- MachineOverview：删除与 MachineOverview 画面上的单元相关联的动态化和图形元素。

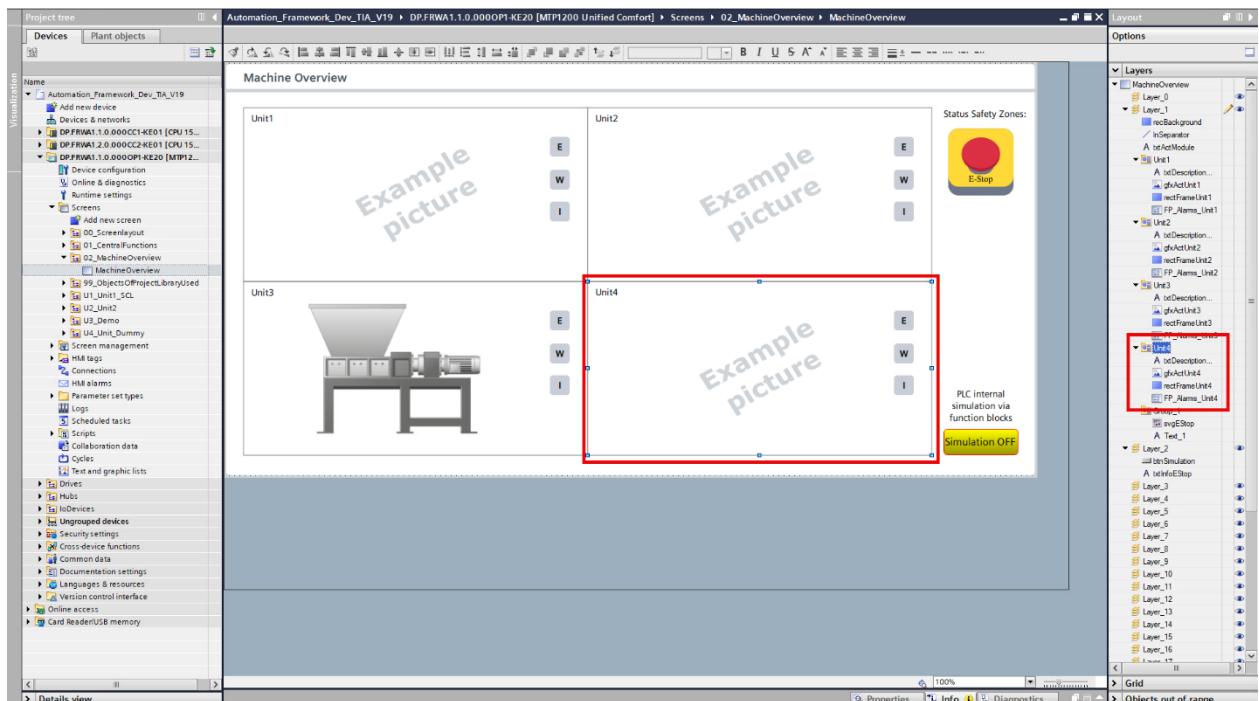


图 9-21 旧的 MachineOverview 画面

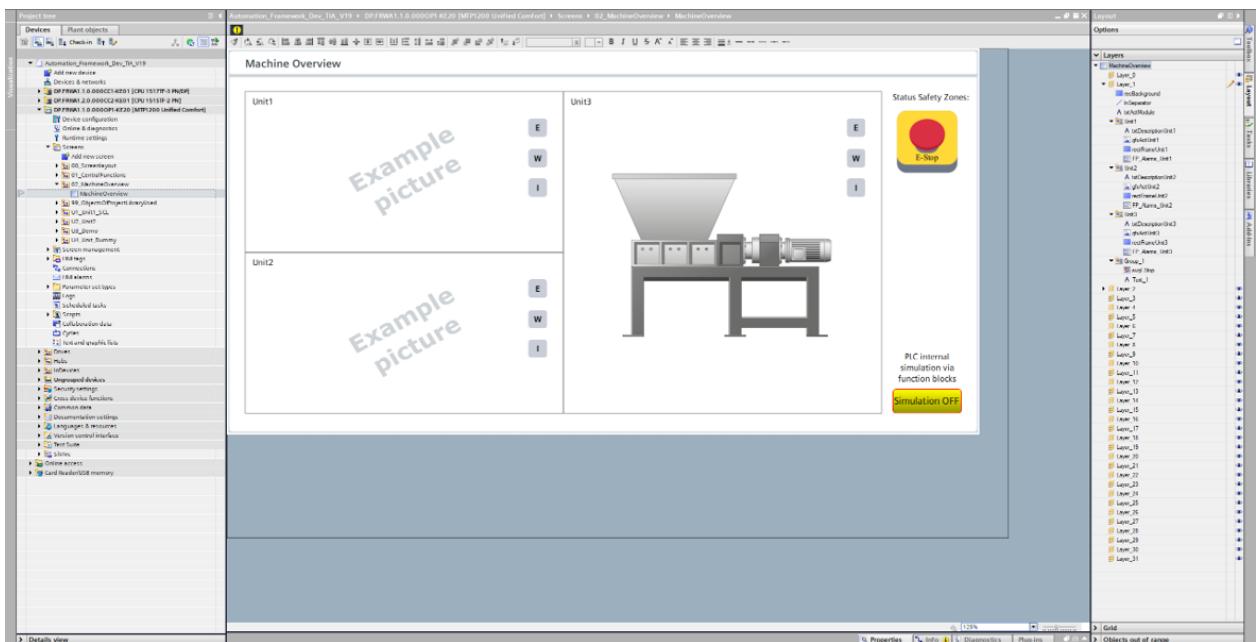


图 9-22 没有单元 4 的新 MachineOverview 画面

- ScreenLayout: 删除 ScreenLayout 画面上引用已删除 Unit 的任何图形元素（按钮、图像等）：

- 在 grpUnits 中，删除 txtUnit4 并调整按钮：

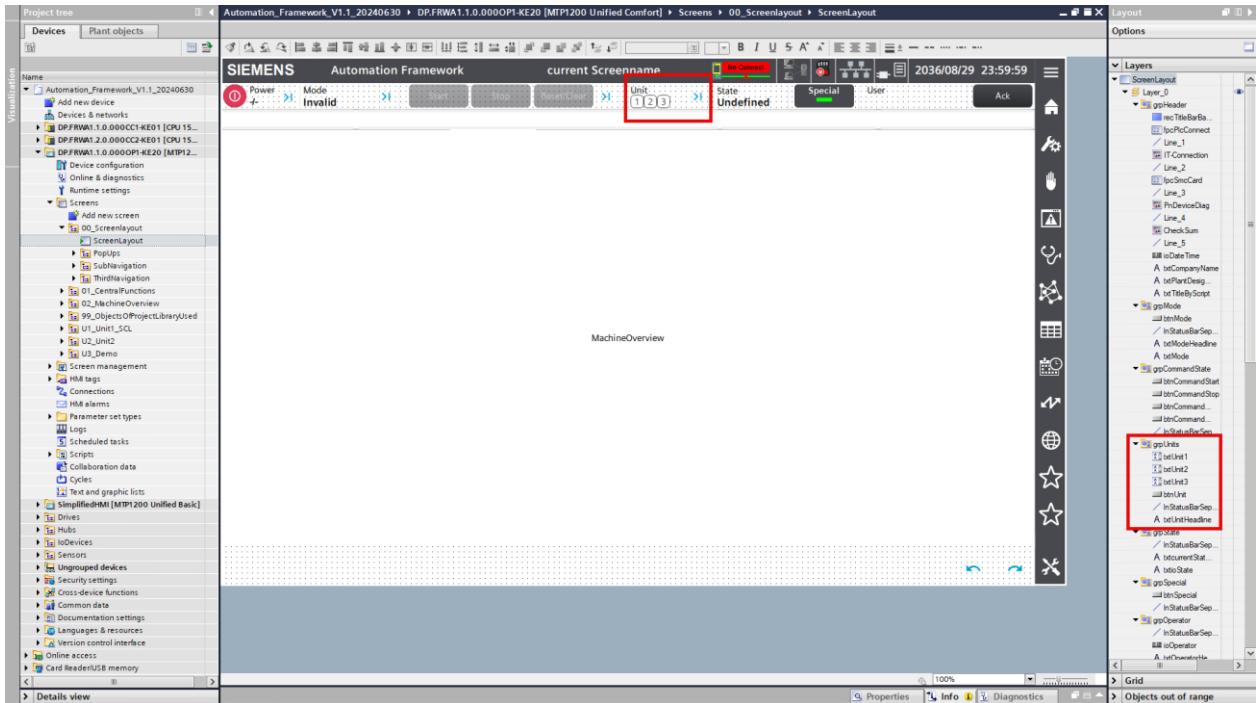


图 9-23 删除单元 4 的 grpUnit

- 在 Template_SelectUnit 中，删除 Unit 4 并重新排列弹出窗口：

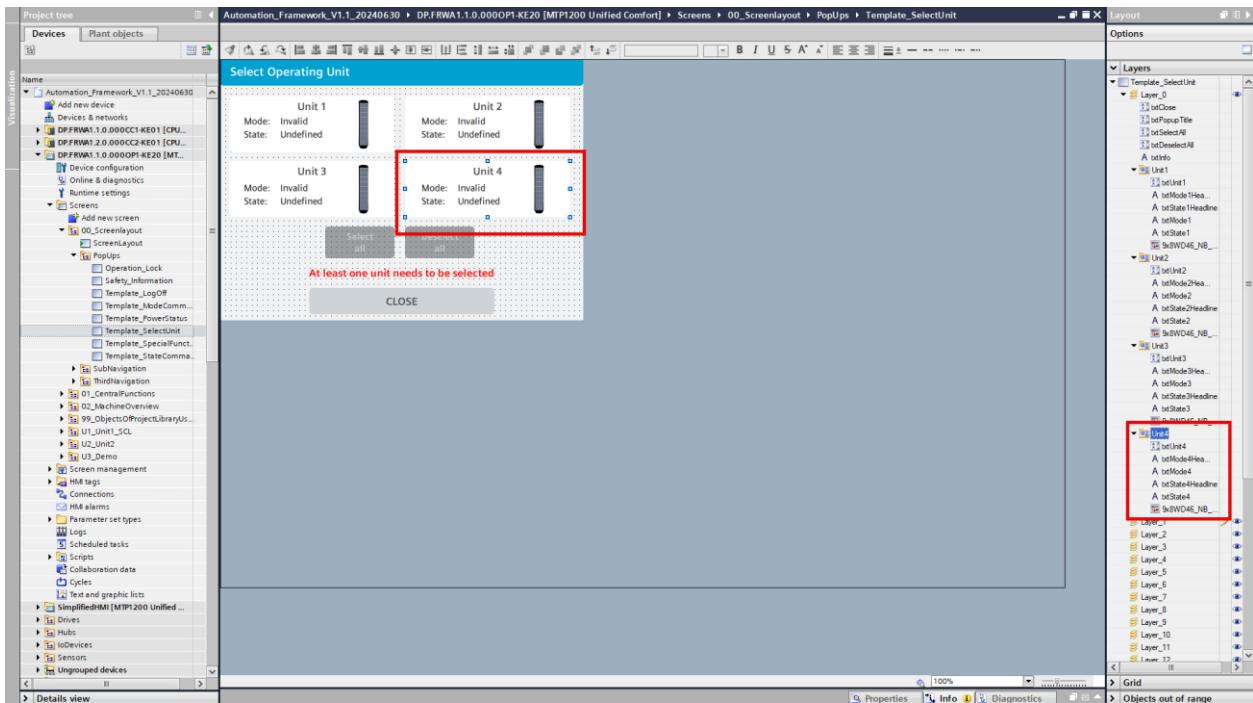


图 9-24 旧模板的 SelectUnit 画面

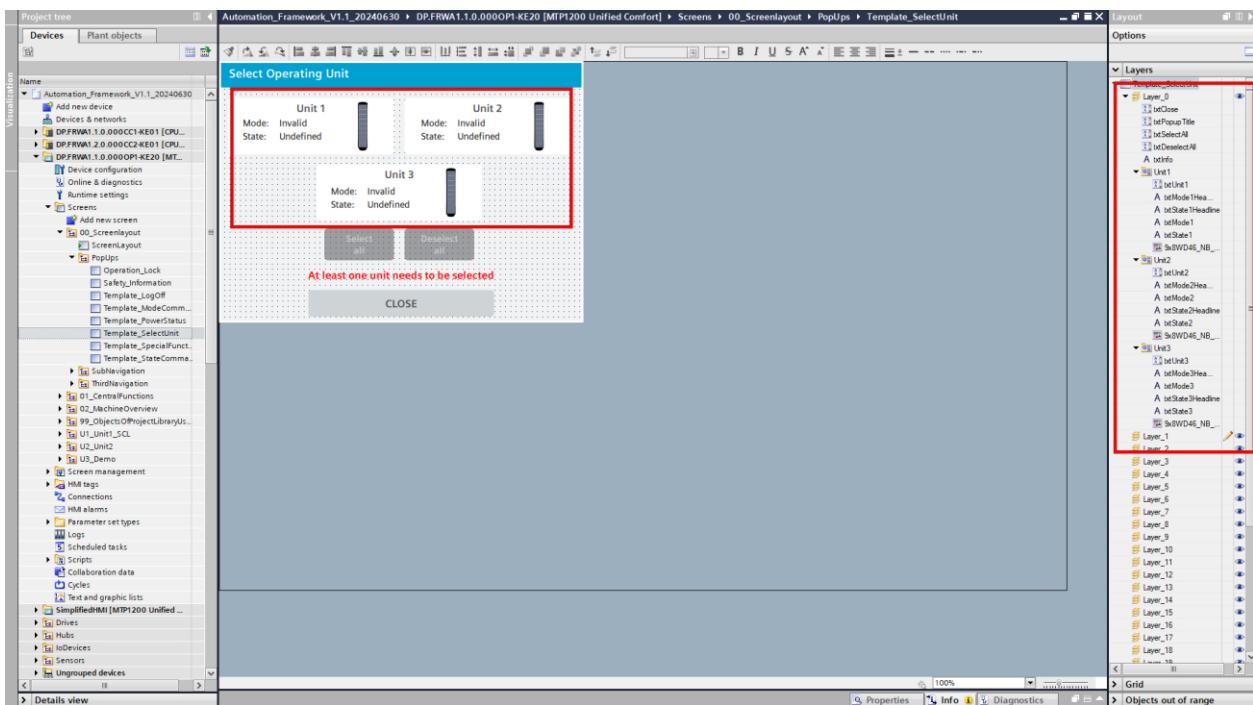


图 9-25 新模板的 SelectUnit 画面

然后，应将导航调整为新的单元数量，删除其他按钮。

- 导航按钮：

- SubNavigation: 删除 SubNavigation 画面上 Unit 的相应按钮。

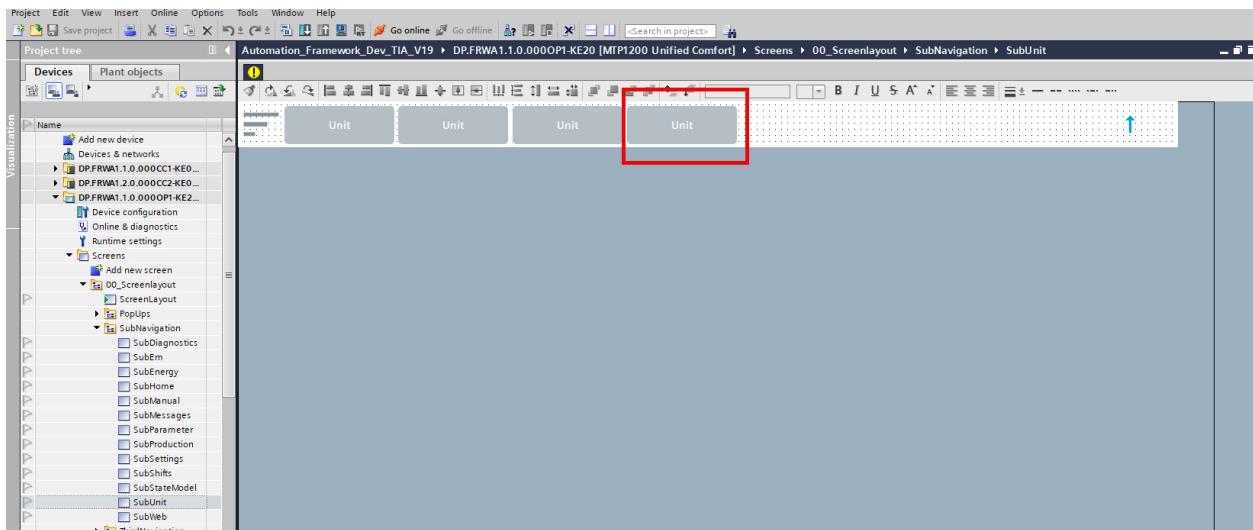


图 9-26 删除 Unit4 的 SubUnit Navigation

在单元的其他按钮中，在脚本中，删除该单元的部分，并使用新的单元数量重新排列：

```

txtSubNav3 [Text box]
Properties Events Texts Expressions

Activated Deactivated Click left mouse button Press key Release key Click right mouse button...
Global definition Asynchronous
1 export async function txtSubNav3_OnTapped(item, x, y, modifiers, trigger) {
2     let screen = "Unit3";
3     let prefix = "U3Em";
4     let count = Tags(screen + "EmNum").Read();
5     Tags.CreateTagSet([{["EM_UnitSelected"], prefix, ["EmCount"], count}, {"SubThirdUnit", 3}, {"SubThirdEm", 0}]).Write();
6     Tags.CreateTagSet([{["selectedUnitDrive"], 3}, {"selectedUnitIdentH", 3}, {"selectedUnitIdentRd", 3}, {"ET200SPMotorstarterInterfaceHMI_ET200SPUnitMotorstarter", 3}]).Write();
7
8
9 HMIRuntime.Tags.SysFct.ResetBitInTag("HmiMainHeader.hmiToPlc.selectedUnits", 0);
10 HMIRuntime.Tags.SysFct.ResetBitInTag("HmiMainHeader.hmiToPlc.selectedUnits", 1);
11 HMIRuntime.Tags.SysFct.SetBitInTag("HmiMainHeader.hmiToPlc.selectedUnits", 2);
12 HMIRuntime.Tags.SysFct.ResetBitInTag("HmiMainHeader.hmiToPlc.selectedUnits", 3);
}

```

图 9-27 Sub-Navigation 的旧脚本

```

txtSubNav3 [Text box]
Properties Events Texts Expressions

Activated Deactivated Click left mouse button Press key Release key Click right mouse button...
Global definition Asynchronous
1 export async function txtSubNav3_OnTapped(item, x, y, modifiers, trigger) {
2     let screen = "Unit3";
3     let prefix = "U3Em";
4     let count = Tags(screen + "EmNum").Read();
5     Tags.CreateTagSet([{["EM_UnitSelected"], prefix, ["EmCount"], count}, {"SubThirdUnit", 3}, {"SubThirdEm", 0}]).Write();
6     Tags.CreateTagSet([{["selectedUnitDrive"], 2}, {"selectedUnitIdentH", 2}, {"selectedUnitIdentRd", 2}, {"ET200SPMotorstarterInterfaceHMI_ET200SPUnitMotorstarter", 2}]).Write();
7
8
9 HMIRuntime.Tags.SysFct.ResetBitInTag("HmiMainHeader.hmiToPlc.selectedUnits", 0);
10 HMIRuntime.Tags.SysFct.ResetBitInTag("HmiMainHeader.hmiToPlc.selectedUnits", 1);
11 HMIRuntime.Tags.SysFct.SetBitInTag("HmiMainHeader.hmiToPlc.selectedUnits", 2);
}

```

图 9-28 Sub-Navigation 的新脚本

- ThirdNavigation: 删除 ThirdNavigationUnit 画面上 Unit 的相应按钮。

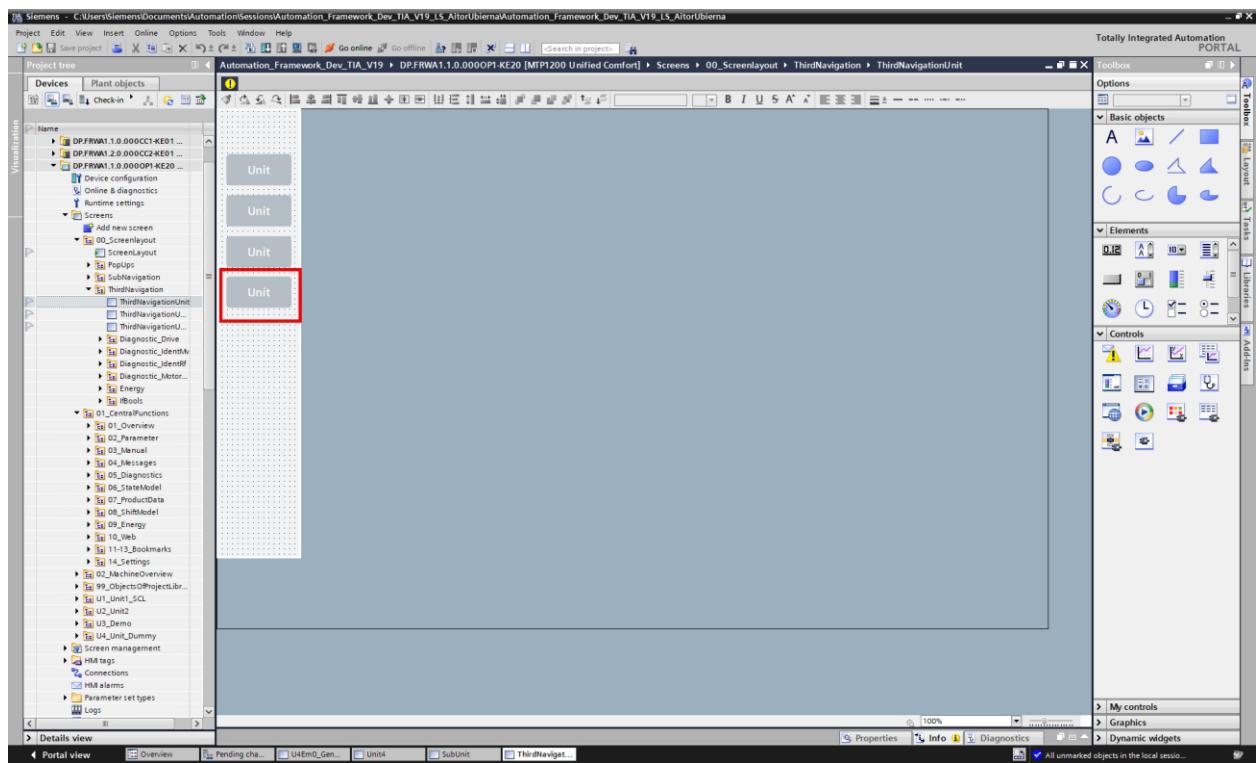


图 9-29 选择要删除的单元

在单元的其他按钮中，在脚本中，删除该单元的部分，并使用新的单元数量重新排列。

旧脚本：

```

txtThirdNav3 [Text box]
Properties Events Texts Expressions
Global definition | Asynchronous | X C ⌂
Activated Deactivated Click left mouse button Press key Release key Click right mouse button
1 export async function txtThirdNav3_OnTapped(item, x, y, modifiers, trigger) {
2 let screen = "Unit3";
3 let prefix = "U3Em";
4 let count = Tags[screen + "Enum"].Read();
5 Tags.CreateTagSet([{EM_UnitSelected: prefix}, [EMCount: count], [SubThirdUnit: 3], [SubThirdEm: 0]]).Write();
6
7 Tags.CreateTagSet([{selectedUnitDrive: 3}, [selectedUnitIdentNm: 3], [selectedUnitIdentRd: 3], [ET200SPMotorstarterInterfaceHMI_ET200SPUnitMotorstarter: 3]]).Write();
8
9 HMIRuntime.Tags.SysFct.ResetBitInTag("HmiMainHeader.hmiToPlc.selectedUnits", 0);
10 HMIRuntime.Tags.SysFct.ResetBitInTag("HmiMainHeader.hmiToPlc.selectedUnits", 1);
11 HMIRuntime.Tags.SysFct.SetBitInTag("HmiMainHeader.hmiToPlc.selectedUnits", 2);
12 HMIRuntime.Tags.SysFct.ResetBitInTag("HmiMainHeader.hmiToPlc.selectedUnits", 3);
13

```

图 9-30 Third navigation 的旧脚本

新脚本：

```

txtThirdNav3 [Text box]
Properties Events Texts Expressions
Global definition | Asynchronous | X C ⌂
Activated Deactivated Click left mouse button Press key Release key Click right mouse button
1 export async function txtThirdNav3_OnTapped(item, x, y, modifiers, trigger) {
2 let screen = "Unit3";
3 let prefix = "U3Em";
4 let count = Tags[screen + "Enum"].Read();
5 Tags.CreateTagSet([{EM_UnitSelected: prefix}, [EMCount: count], [SubThirdUnit: 3], [SubThirdEm: 0]]).Write();
6
7 Tags.CreateTagSet([{selectedUnitDrive: 3}, [selectedUnitIdentNm: 3], [selectedUnitIdentRd: 3], [ET200SPMotorstarterInterfaceHMI_ET200SPUnitMotorstarter: 3]]).Write();
8
9 HMIRuntime.Tags.SysFct.ResetBitInTag("HmiMainHeader.hmiToPlc.selectedUnits", 0);
10 HMIRuntime.Tags.SysFct.ResetBitInTag("HmiMainHeader.hmiToPlc.selectedUnits", 1);
11 HMIRuntime.Tags.SysFct.SetBitInTag("HmiMainHeader.hmiToPlc.selectedUnits", 2);

```

图 9-31 Third navigation 的新脚本

调整导航后，应删除设备模块所涉及的文本列表（在本例中为“U4Em”），并从用于单元（称为“Units”）的文本列表中删除单元名称。

The screenshot shows the SIMATIC Manager software interface. On the left, there's a navigation tree under the 'Visualization' tab. In the center, there's a table titled 'Text lists' with columns for Name, Selection, and Comment. At the bottom, there's a smaller table titled 'Text list entries' with columns for Default, Value, and Text. The row 'U4Em' in the main table is highlighted with a red box.

Name	Selection	Comment
14_Settings	Value/Range	
02_MachineOverview	Value/Range	
99_ObjectsOfProjectLibr...	Value/Range	
U1_Unit1_SCL	Value/Range	
U2_Unit2	Value/Range	
U3_Demo	Value/Range	
U4_Unit_Dummy	Value/Range	
Screen management		
HMI tags		
Connections		
HMI alarms		
Parameter set types		
Logs		
Scheduled tasks		
Scripts		
Collaboration data		
Cycles		
Text and graphic lists		
SimplifiedHMI [MTP1200 Unif...		
Drives		
Hubs		
IoDevices		
Sensors		
Ungrouped devices		

...	Default	Value	Text
1	<input checked="" type="radio"/>	1	General
		<Add new>	

图 9-32 必须删除的文本列表的位置

- 编译：编译项目以检测并解决由删除引起的任何错误。
- 仿真：仿真项目以验证单元是否已正确删除，其余功能是否按预期工作。

9.9. 如何调整 HMI 的大小

要将 HMI 的大小调整为所需的面板大小，您应该按照以下步骤操作：

1. 更改目标设备

- a. 导航到项目中的设备设置。
- b. 选择与您要使用的屏幕分辨率或显示大小相匹配的所需设备。

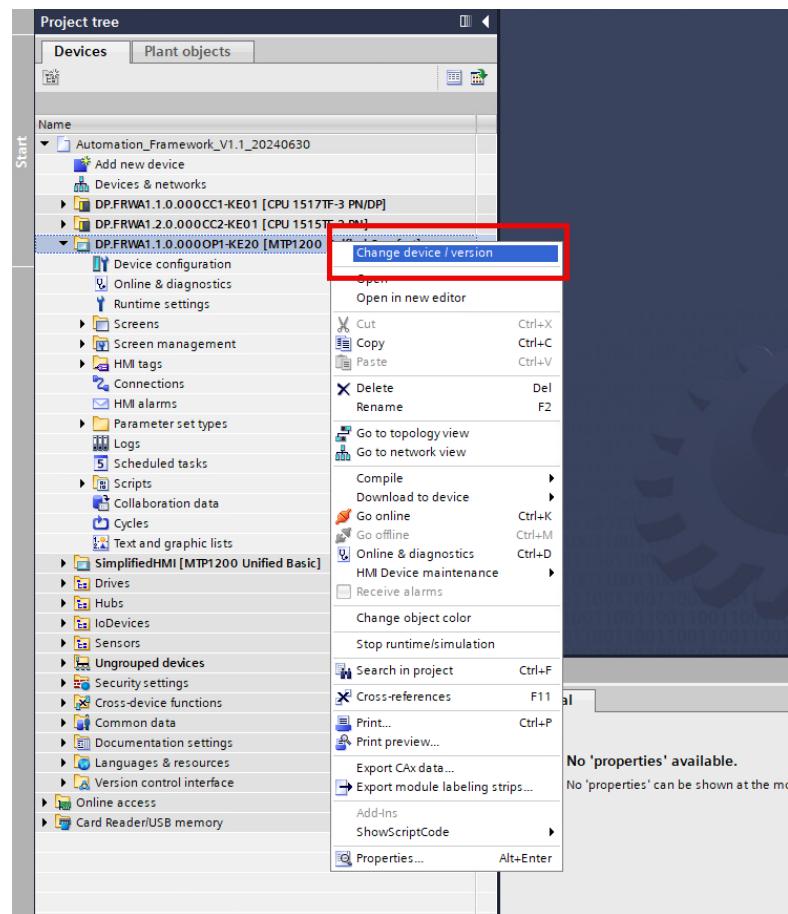


图 9-33 选择 HMI 的“Change device/version”属性

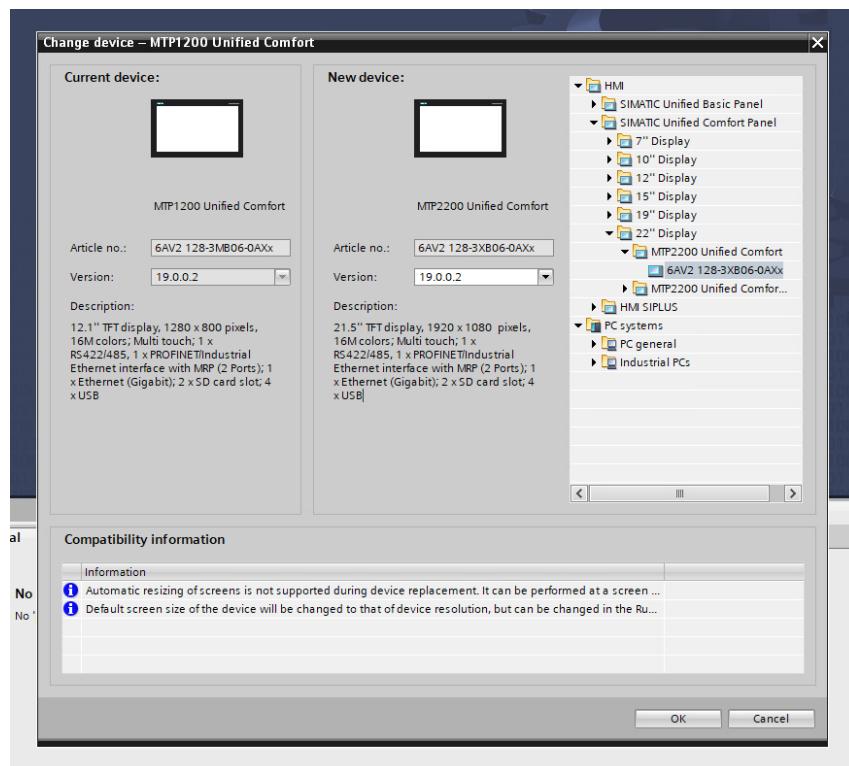


图 9-34 选择所需的设备调整大小

2. 调整画面大小以匹配显示

选择目标设备后，您可以继续调整画面大小以适应此新的显示大小。请执行以下步骤：

- 在项目树中，找到要调整大小的画面。
- 右键单击画面，然后从上下文菜单中选择“Resize to Display”选项：

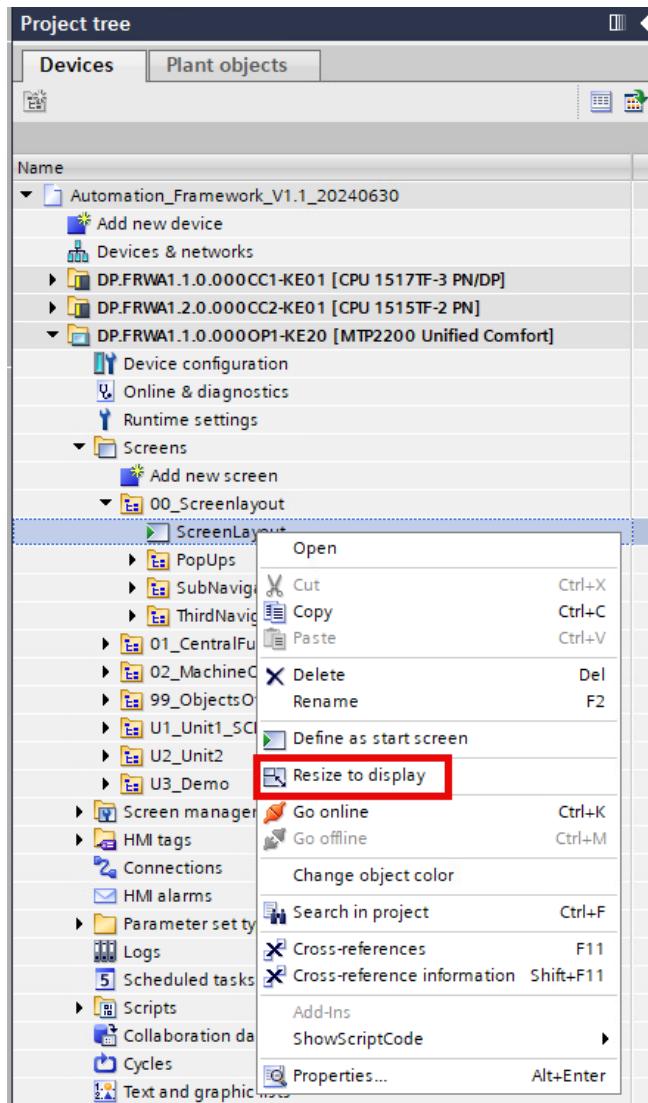


图 9-35 “Resize to display” 属性

此选项会自动调整画面尺寸的大小以匹配所选设备的画面尺寸。画面上的大多数对象都将适当调整大小，从而确保不同分辨率下的外观一致。

3. 调整面板容器

虽然画面和大多数对象可以正确调整大小，但某些元素（如面板）可能需要额外调整：

- 面板内的容器会自动调整大小，但是，面板本身的大小可能无法调整。
- 要调整面板内内容的大小，请导航到面板的属性面板。
- 找到标有“使屏幕适合窗口”的选项并启用它。这将确保面板的内容在调整大小的屏幕中正确缩放。

Properties			Events	Texts	Expressions
Name	Static value	Dynamization (1)			
Appearance					
Appearance - style item	HmiFaceplateContainer				
Focus - show visual		<input checked="" type="checkbox"/> None			
Title row - color	145, 147, 154				
Window settings	None				
Format					
Horizontal scroll bar - position	0				
Horizontal scroll bar - visibility	Collapsed				
Size - fit	Fit screen to window	<input type="checkbox"/> None			
Vertical scroll bar - position	0				
Vertical scroll bar - visibility	Collapsed				
Zoom - allow		<input type="checkbox"/> None			
Zoom - factor	1				
Miscellaneous					
Connection status	None				
Faceplate type	LAF_fpManageAlarms V 1.1.0				

图 9-36 面板属性中的“Fit to screen window”属性

4. 管理静态尺寸值

某些对象或元素可能定义了静态尺寸值，这可能会阻止它们自动调整大小。一个常见的示例是具有固定宽度的导航栏：

- 标识具有静态大小值（如 width）或 height 的任何对象，尤其是在导航元素中。
- 删除这些静态值以允许对象根据屏幕尺寸动态调整大小。
- 删除静态值后，对象应自动调整以适应新的屏幕大小。

```

Properties Events Texts Expressions
JS Global definition Asynchronous X C G
Click left mouse button
Click right mouse button
Loaded Cleared
1 export async function FMeterPOS_OnLoaded(item) {
2 //HMIRuntime.UI.FindItem("~/swThirdNavigation").Width = 130;
3
4 module_navigation.ShowTitle();
5
6 module_colors.ScreensBackColors();
7
8 HMIRuntime.UI.SysFct.ChangeScreen("ThirdEnergy", "~/swThirdNavigation");
9
10 module_navigation.ShowThirdNavigation();
11 HMIRuntime.Tags.SysFct.SetTagValue("SubEnergy", 4);
12
13 }

```

图 9-37 在脚本中删除了静态值

5. 最终调整和测试

完成调整大小操作后，建议在目标设备上查看和测试画面。确保所有对象、文本和导航元素都得到适当的缩放，并且用户界面保持功能和视觉吸引力。

10. 设备模块

10.1. 常规结构

EM 始终由一个软件单元组成，是命令和数据流结构的一部分，也是[第 6 章](#)中介绍的 HMI 连接概念的一部分。

EM 包括机器的特定 CM，这些 CM 执行设备模块指定的功能。

主程序结构

- 在 EM 级别上，首先评估模块本身与单元的当前关联状态。在[与单元链接](#)一章中将讨论将一个 EM 与单元关联和取消关联的含义。
- 接下来，处理代表传感器的控制模块块，以获得设备模块的过程输入状态。
- 接下来，EM 将单元的不同状态映射到用户程序或序列的不同行为，应该提供模块的主要功能。因此，可以将这些功能划分为作业控制概念，这将在[作业控制](#)一章中进一步介绍。
- 在调用 CM 之前，对是否以及何时复位和激活代表 Actuator 的 CM 进行一般评估。
- 然后，将构建当前状态的完成反馈和对单元的命令。
- 最后，处理模块的警报。

信号流

图 10-1 设备模块中的信号流显示了 EM 内信号（命令和状态）的互连。

常规模式和状态来自单元级别。这决定了设备级别是在自动模式、维护模式、手动模式还是任何其他用户自定义的模式下运行。如果启用了手动模式，则允许通过 HMI 手动操作控制模块、作业选择和序列。DB HmiInterface 连接到 HMI 画面中的相应面板。提供的示例序列允许手动连续操作以及分步模式。

单元的模式和状态通过 FC “LUC_InterlinkToParent” 和 “LAF_ControlEm” 进行评估。根据设备模块的链接状态，要么在 EM 中使用单元信号，要么在未链接的情况下使用 HmiInterface 或 ControlNodes 信号。FB CallEquipment 的代码与特定客户密切相关。所示的实现只是示例。在 AF 中，显示了具有不同实现方式的多个 Em。也就是说，可以根据配方或作业调用不同的序列。序列可以在 LAD/FBD、SCL 和 GRAPH 中编程。

该序列使用以下中央数据块与控制模块进行通信：DB ControlNodes。具有高透明度和灵活性的优势。不建议在控制模块或序列之间创建许多直接互连，而是使用 DB ControlNodes 作为过程命令和状态的中央数据中心。任何客户特定的控制模块块都可以以某种方式进行编程，以便将命令和状态放置在 DB ControlNodes 中。

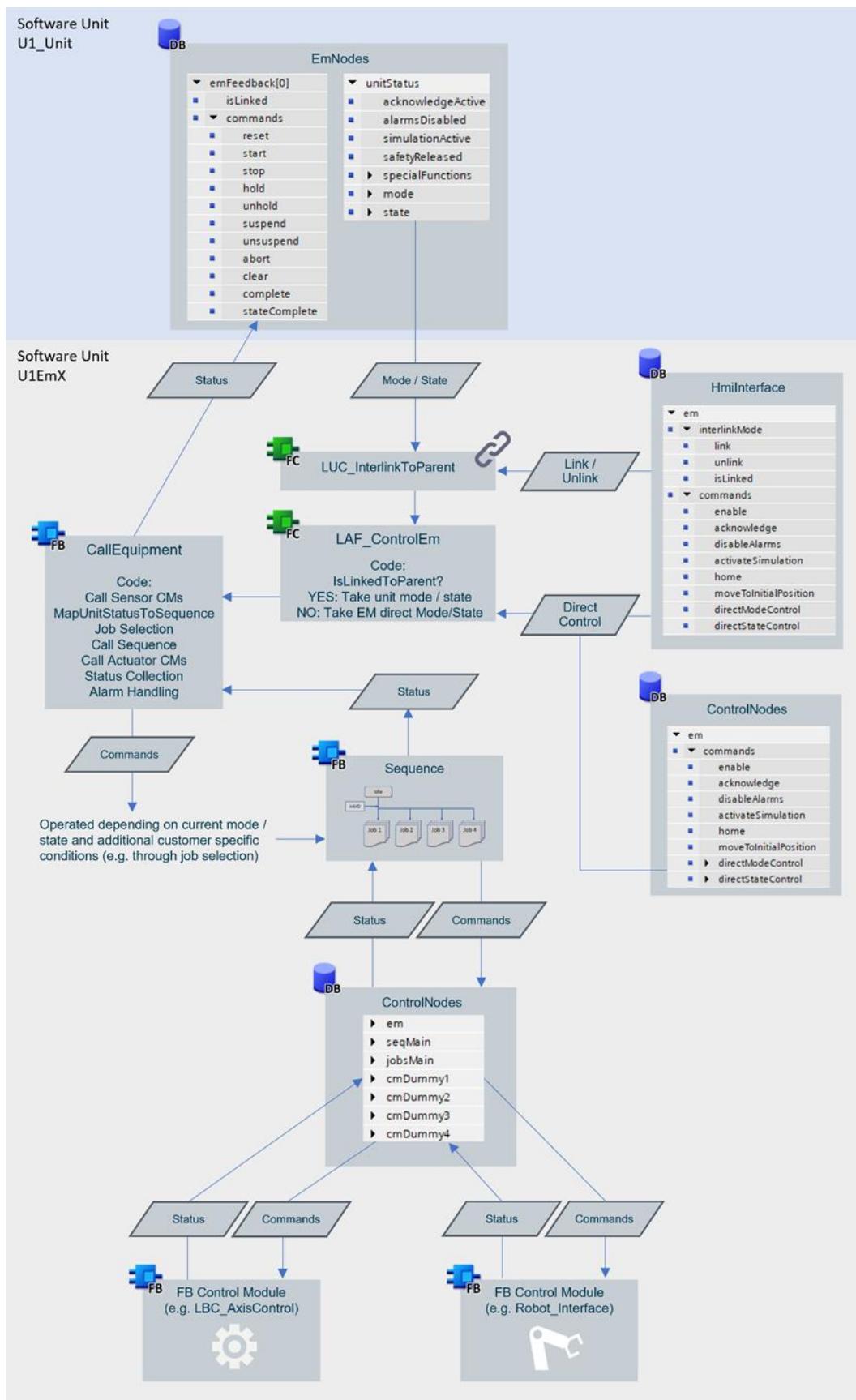


图 10-1 设备模块中的信号流

10.2. 接口和块

设备模块的主要职责包括：

- 将过程逻辑嵌入到系统结构中
- 硬件级别的接口

为了实现这些功能，在代表设备模块的软件单元中提供了以下设置和块：

关系

每个 EM 都需要与 ObjectsOfProjectLibraryUsed 和关联单元建立关系。此外，EM 关系可以包括在模块内处理的工艺对象。下图显示了演示 EM 中的高性能运动，其中包括讨论的关系。

Relations		
Selected unit	Relation type	Accessible element
▼ U3Em3_HighPerfMotion		
U3Em3_HighPerfMotion	► Software unit	ObjectsOfProjectLibraryUsed
U3Em3_HighPerfMotion	► Software unit	U3_Demo_____...
U3Em3_HighPerfMotion	► Technology object	U3Em3_PositioningAxis
U3Em3_HighPerfMotion	► Technology object	U3Em3_SynchronousAxis

图 10-2 设备模块关系图

程序块

同样在设备模块级别，通用程序架构基于[程序结构](#)一章中的描述。为了阐明各个块在设备模块级别是如何工作的，以下部分将更深入地介绍 EM 的各个程序块。

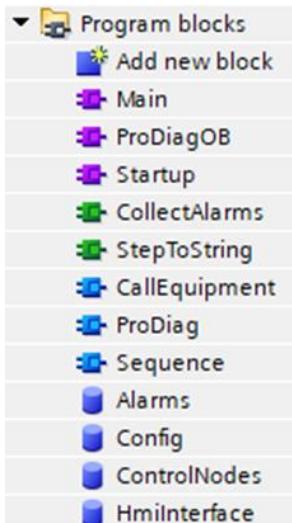


图 10-3 设备模块的程序块

OB Startup (Em)

在 EM 的 OB Startup 中，设置了参考指示符和轴数据的组态。

FC StepToString

StepToString 函数将 EM 中运行的序列的整数值转换为可以在 HMI 上显示的字符串值。这增加了可读性。必须根据定义的步骤和步骤名称为每个序列调整函数。

FB Sequence

如前所述，EM 级别的核心功能是嵌入过程逻辑。FB Sequence 管理模块的顺序逻辑。

DB Config (EM)

数据块“Config”包含 EM 和底层模块的所有组态数据。这包括 em 参考标识符、轴和控制模块的组态数据。组态通常在调试期间设置一次，在机器运行期间不更改。通过保存此数据块（例如导出），EM 的组态已被存储，并且可以轻松导入到新的 PLC 中。

DB ControlNodes (EM)

DB ControlNodes 保存 EM 本身的运行系统数据、序列以及用于控制和监视 CM 作业数据和变量。

DB HmiInterface

在 EM 级别，所有进出 HMI 的命令都通过 DB HmiInterface 传递。它包含的对象与存储在 DB ControlNodes 中的对象相同，但为这些对象提供 HMI 接口值。

FB CallEquipment 执行以下操作



图 10-4 基于 Em3 示例的 FB CallEquipment 内容

将 EM 互连到单元/控制 EM

在使用当前单元模式和状态之前，EM 会评估它当前是否关联到单元，或者是否在 HMI 上设置了取消链接的命令。这由 LUC_InterlinkToParent 块处理。如果 EM 未关联，则可以使用 LAF_ControlEm 块仿真单元模式和状态。

调用 CM (传感器)

这些程序段正在调用评估设备模块的传感器值的 CM。传感器和执行器是分开的，以便在 PLC 循环中始终具有最新值。评估实际输入值，然后在程序中处理并用其更新执行器的 CM。

将 Unit Status 映射到 Sequence Status

此程序段调用一个函数以根据当前单元模式和状态为作业和设备模块的序列设置前提条件。例如，当启用单元的手动模式时，作业和序列管理也设置为手动模式。此外，该模块具有 input bits，用于定义不同单元状态下作业管理的行为。因此，每个

输入都可以配置当设备更改为特定状态时是否应中断当前作业。如果输入信号为 true，则当特定设备状态被激活时，作业将中断。

选择作业

为了分离 EM 的功能，添加了作业管理。然后，通过多个 Select Job 块，在运行期间触发函数。在[作业控制](#)一章中将进行详细说明。

EM 初始化

该程序段用于展示如何处理设备模块的状态变量。还有一些额外的 status bits 可以在这个程序段或其他程序段中处理。在此示例中，如果初始化作业已完成，则 EM 将设置为初始化。其他作业可以将该状态作为其自身执行的前提条件。

调用序列

在评估应执行的作业之后，将调用 Sequence。在本例中为 S7-Graph 序列。步序列是根据作业管理来构建的。

复位 CM / 启用/禁用 CM

在调用 CM 之前，对是否以及何时复位、启用和禁用 CM 进行一般评估。

调用 CM（执行器）

如前所述，在处理步骤序列之后，使用特定的 CM 评估执行器的输出。

确定生产/手动模式中的 StateComplete (SC)/StateComplete (SC) 摘要

在处理完过程逻辑和硬件后，输出来自 EM 的反馈。这是通过分析在生产模式还是手动模式下满足条件以向单元发送完成状态来完成的。

收集报警

在收集报警功能中，可以将任何事件分配给报警。ProDiag 可以监控此特定报警，并将其映射到设备的响应（例如中止、停止、保持）。有关 AF 中过程诊断的更多信息，请参见“[诊断原理](#)”一章。

管理报警

此块从 ProDiag 监控派生命令。ProDiag 监控包含定义这些监控响应的类别（例如，中止、停止、保持）。它们将作为命令移交给状态收集器。

10.3. Hmi 画面

设备画面可由用户自由编程，并取决于机器。在本演示中，实现了 Em Hopper、Em HighPerformanceMotion 和 Em ConveyorStation 的简单示例。

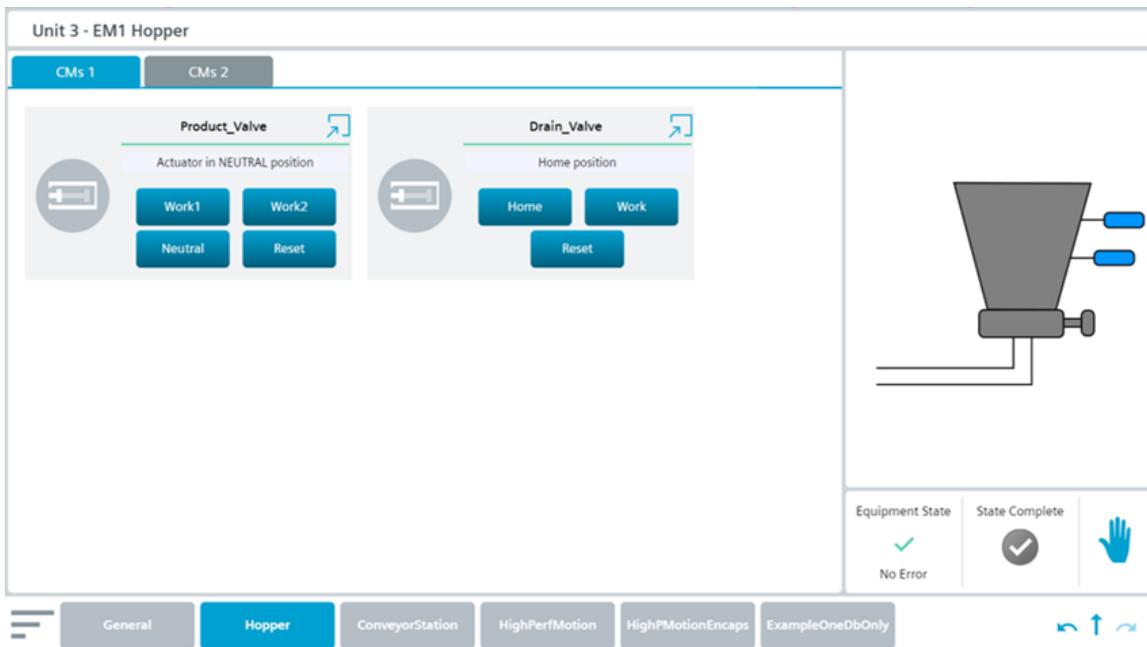


图 10-5 EmHopper 的设备画面

设备画面包含以下面板，因此能够手动控制和监控：

- 作业
- 序列
- 关联状态，如果未关联，则为仿真单元模式和状态
- 控制模块

在此示例中，使用 LBC 库的面板。当单元状态机处于手动模式时，所有面板都已启用。用于控制序列、作业和关联状态的面板可以通过按下右下角的手形符号来访问。将打开一个弹出窗口，如下图所示：

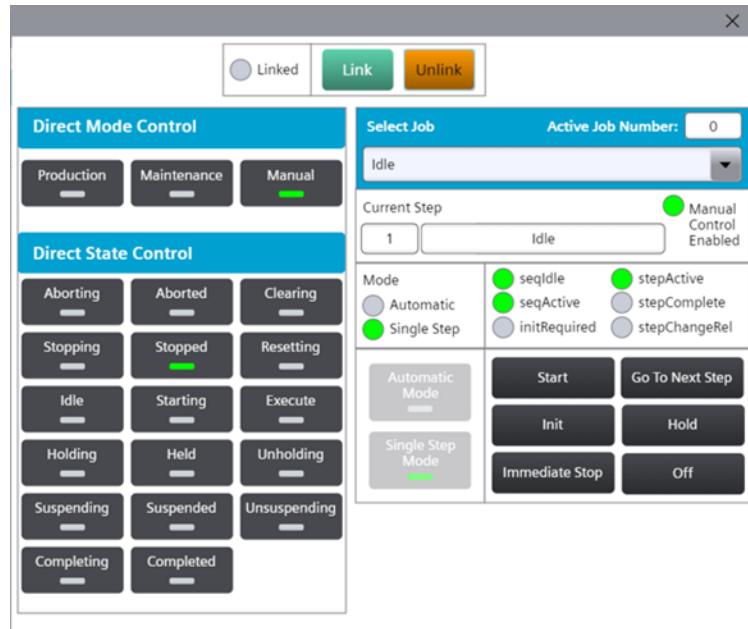


图 10-6 EM 的手动控制面板

10.4. 与单元链接

AF 可以链接和取消链接已分配单元的设备模块。这在调试期间非常有用，此时尚未集成所有 EM。另一个用例是，某些机器可能具有冗余的 EM，并且机器提供了关闭一些 EM 的可能性，用于维护或由于所需的吞吐量较低，例如冗余进料系统。

如果 EM 链接到设备，则 EM 将应用设备模式和状态并跟随设备。

如果 EM 未链接到设备，则 EM 不会跟随设备，也不会应用设备模式和状态。它与设备断开连接，可以直接操作。如果 EM 未链接，则设备将不会考虑 EM 反馈。设备将忽略 EM。

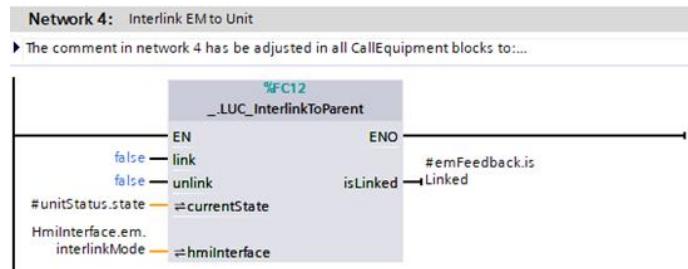


图 10-7 EM 的链接功能

FC LUC_InterlinkToParent 允许 EM 使用输入“link”和“unlink”在内部链接或取消与设备的链接。此外，操作员还可以通过 hmiInterface 建立或删除连接。只有当父级（包含相应设备模块的设备）处于 Aborted、Clearing 或 Stopped 的活动状态时，才能进行链接或取消链接。

将设备模块链接到单元：

启用手动模式时，将逻辑连接到输入链接，或从 HMI 发送链接命令。

取消设备模块与单元的链接：

启用手动模式时，将逻辑连接到输入取消链接，或从 HMI 发送取消链接命令。

10.5. 直接控制

块“LAF_ControlEM”评估 EM 是否链接到父级。如果已链接，则采用“unitStatus”输入并将值复制到输出“unitStatusUsed”。该变量用于 EM 的进一步逻辑中。因此，如果 EM 已链接，它会将设备状态完全应用于 EM。

如果 EM 未链接到设备，则只有 unitStatus 中的 acknowledge 和 safetyReleased 应用于 EM。模式和状态可以由“localInterface”直接控制，也可以由操作员通过“hmiInterface”输入输出引脚控制。这意味着任何模式和状态都可以被赋给 EM。EM 中没有状态机，因此无法实现自动模式或状态转换，状态的执行顺序是完全灵活的。

通过直接 EM 控制，可以轻松测试 EM 是否在相应状态下执行了正确的工作和任务。例如，您可以使用 resetting 状态并测试在该状态下执行的逻辑。

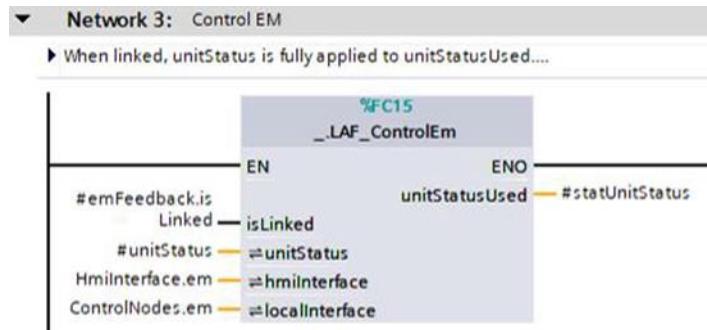


图 10-8 EM 的直接控制功能

10.6. EM – EM 关联

由于单元的 EM 与过程密切相关，因此可以实现 EM 间的内部信号交换。该单元的 DB “EmNodes” 在这里作为中心数据点。各个 EM 之间的接口可以集成到 emRelations 数据结构中。下图显示了 emRelations 的演示实现。

emRelations		Struct
■	EM1	Struct
■	readyToReceive	Bool
■	readyToDeliver	Bool
■	EM2	Struct
■	EM3	Struct

图 10-9 EM 关系信号的演示实现

此示例结构用于从位于其他两个 EM 中间的 EM 传输信号。它从第一个 EM 接收工件，并将它们交付给第三个 EM。根据状态，它会设置 readyToDeliver 或 readyToReceive 位。因此，相邻 EM 可以对这些位做出响应并交付或接收工件。

EM 信号交换可用于按顺序执行 EM 的不同功能。这种互连可以用来实现单元的一个过程。

10.7. 作业控制

为了实现 EM 的模块化特性，设备可以提供的不同功能被划分为不同的作业。例如，一个作业可以是 EM 的初始化，也可以是生产过程的一部分，例如将生产的对象运输到下一个 EM。这种分离的处理由“LAF_SelectJob”块进行。它提供了对 EM 可以执行的一个作业的控制。因此，为 EM 定义的每个作业都必须具有其“LAF_SelectJob”实例和 EM 的用户常量中的特定数字。

通过使用“LAF_SelectJob”块的 startCondition 输入激活 Job，分配的编号将被转移到 EM 的序列中，并且作业的分支将启动。然后，每个作业都包含实现应执行的特定功能的所有步骤。因此，CM 交互将发生在不同作业的这些步骤中。

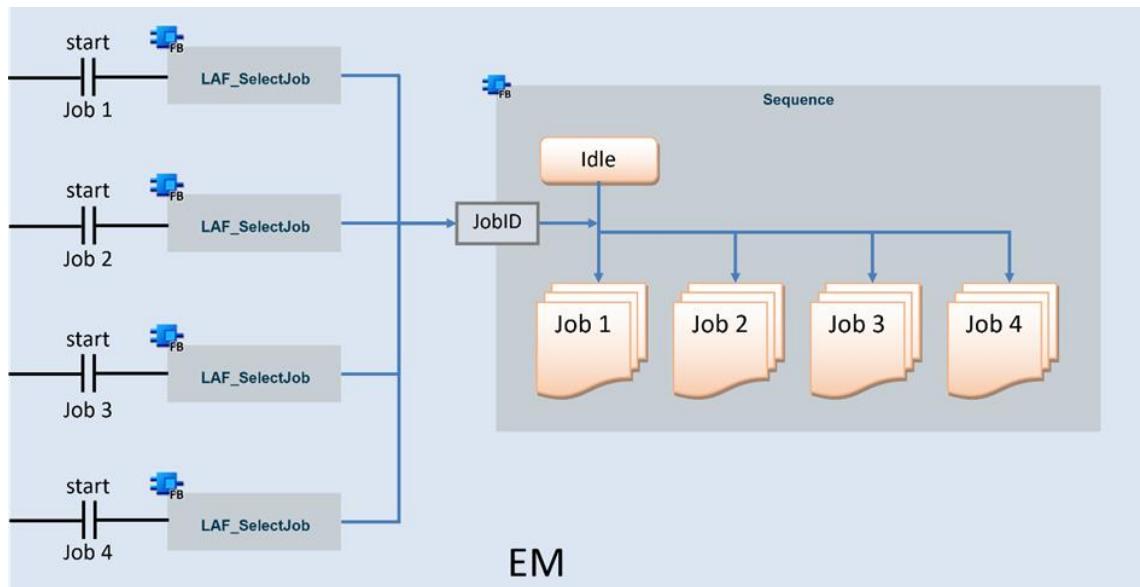


图 10-10 Em 的作业管理

使用块时，定义的作业编号必须关联到实例，并且必须定义作业是否为 singleCycleJob（True = 当满足 startCondition 时，作业执行一次，False = 当满足 startCondition 输入的条件时，作业循环执行）。此外，可以连接 EM 的“parentReferenceDesignator”以将块包含在 AF 的诊断概念中。

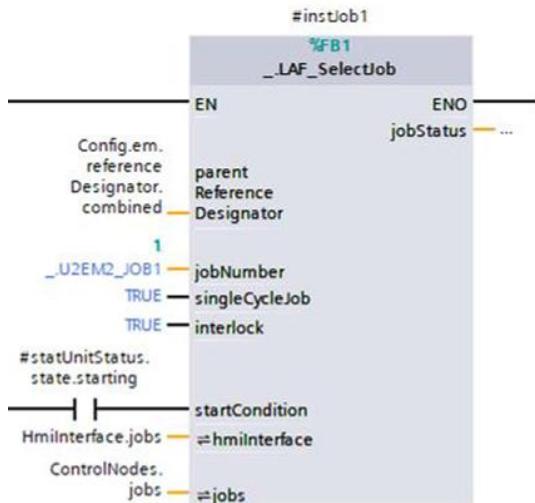


图 10-11 FB“LAF_SelectJob”的调用

如果不满足条件，则“interlock”输入会阻止特定作业的执行。因此，输入必须始终保持“True”才能执行作业。如前所述，“startCondition”输入定义了执行作业必须满足的条件。

“hmiInterface”的 InOut 应包含建立块与 HMI 之间连接所需的数据以进行控制。“jobs”的 InOut 则会将数据传送到 PLC 内的特定点，以共享作业信息。此外，作业的当前状态可通过输出信号“jobStatus”获取。它包含“LAF_typeJobStatus”UDT，该 UDT 提供作业是否被锁定、是否处于活动状态、是否已完成或在执行过程中是否被中止的信息。因此，这些信号可被链接以用于进一步处理，例如，用于反馈特定单元状态的“stateComplete”条件。

如果当前设备模式是维护或手动模式，则“LAF_SelectJob”块也将更改为手动行为。该块现在通过“hmiInterface”接受作业编号，并将信息转发到“jobs”接口。作业的状态仍将被处理并可以监控。

10.8. 顺控

AF 中有三种不同类型的顺控可用。根据要求，可以选择 S7-GRAFH、LAD 或 SCL 序列。所有顺控都至少使用一个模板和一个示例实现。下一章将解释这些实现之间的区别。可以在下面显示的软件单元中找到实现。

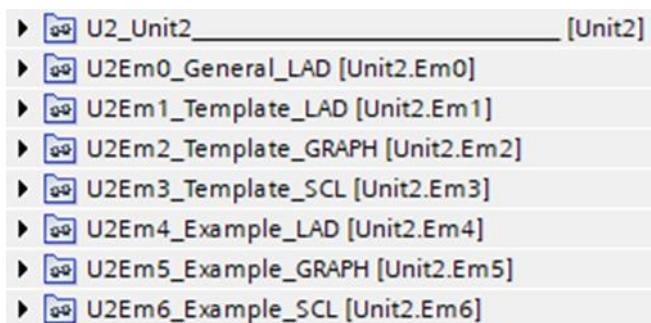


图 10-12 包含模板和示例 EM 的软件单元

各种 EM 的结构都基于本章前面给出的解释。唯一的区别是 Sequencer 和 Jobs。每个顺控都遵循[作业控制](#)一章中介绍的概念，其中 EM 的各个功能被划分为 jobs。为了提供序列的常规功能，并在不进行进一步的 EM 级编程工作的情况下实现架构，AF 为上述三种编程语言中的每一种都提供了使用控制和监视 FB 的可能性。

S7-GRAFH

在 Graph Sequencer 内部，提供的两个 FB 将序列控制变量关联到 Graph FB 的内部命令，称为“RT_DATA.MOP”。第一个功能“LAF_ControlGraphSequence”被添加到永久预指令程序段中，并设置序列模式，初始化、保持顺控并将其更改为单步模式。此外，该功能还会评估当前作业是否被取消，这一信息将在“resetJobs”输出中显示，并可用于重置“LAF_SelectJob”块的状态位。下图展示了该块的调用，以及必须建立的连接，以便将数据块“HmiInterface”和“ControlNodes”的信号映射到功能块，进而映射到“rtDataMop”的 InOut。

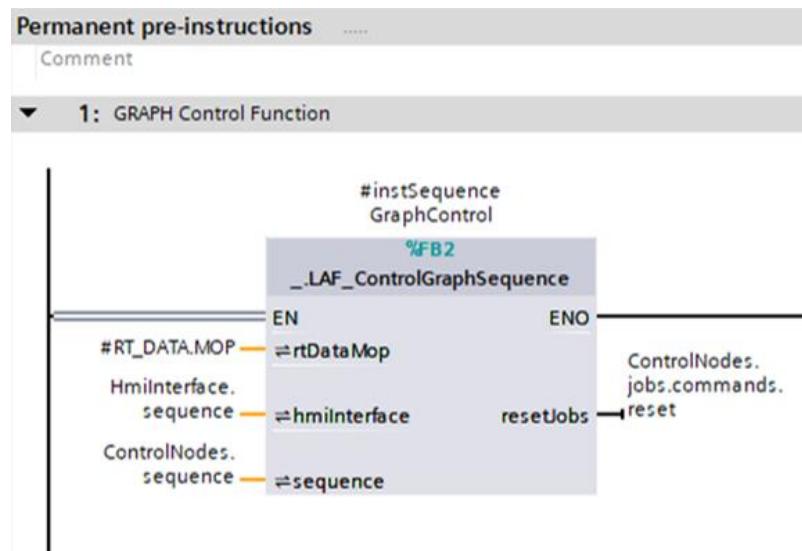


图 10-13 调用 FB“LAF_ControlGraphSequence”

第二个功能块“LAF_StatusGraphSequence”在永久的后指令部分调用，并设置序列的当前状态。因此，内部 Graph Signals 用于映射到 DB “ControlNodes” 和 DB “HmiInterface” 信号。同时，“setJobNumber”输入值被用来设置接口信号中的活动任务。为了评估序列器是否处于空闲或停止的所需步骤中，需要分配 S7-Graph 的步骤内部变量。为了将该功能块纳入自动化功能（AF）的诊断概念中，EM 的组合引用指示符可以连接到功能块的“parentReferenceDesignator”输入。块调用和相应的接口分配如图所示。

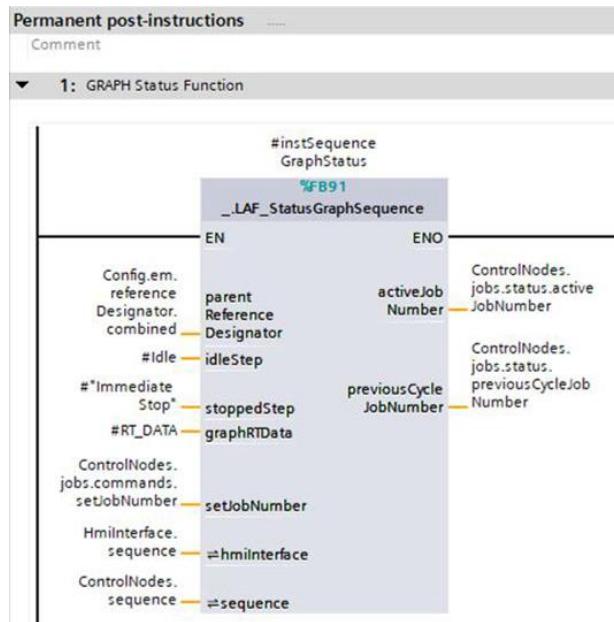


图 10-14 调用 FB“LAF_StatusGraphSequenc”

LAD

功能块“LAF_ControlLadSequence”和“LAF_StatusLadSequence”与 Graph Sequence 中使用的块的区别在于，无需将 DB “ControlNodes” 和 DB “HmiInterface” 信号映射到顺控的内部数据结构。这具有一个特征，即 Idle 和 Stopped 步也必须在“LAF_ControlLadSequence”的接口中指定。其余的功能和行为类似于 graph 序列的处理。两个块的正确连接如下图所示。

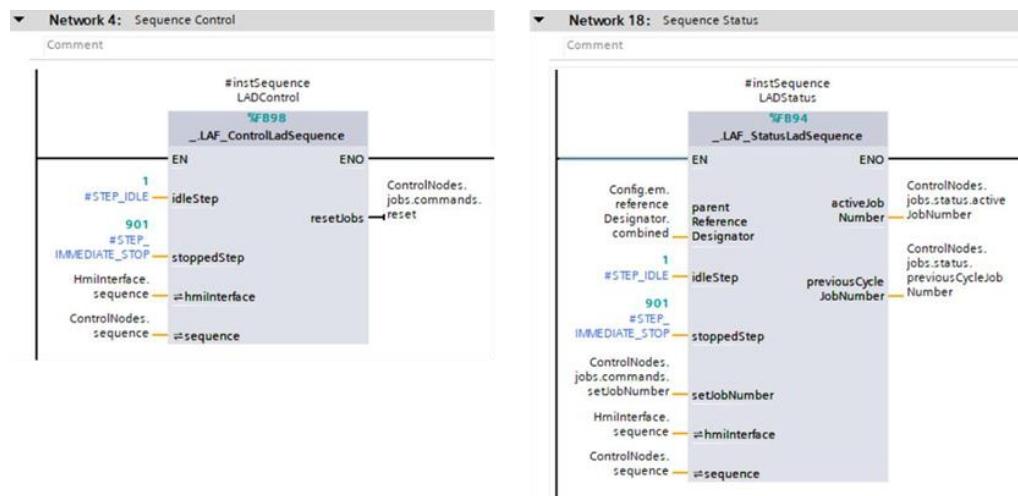


图 10-15 调用 FB“LAF_ControlLadSequence”和“LAF_StatusLadSequence”

SCL

序列和作业处理的概念也涵盖了 SCL 编程语言。LAD 和 SCL 的实现仅在控制和监视块中的一些内部细节上有所不同。对于 SCL，“空闲”和“已停止”步也必须分配给 FB “LAF_ControlSclSequence” 和 “LAF_StatusSclSequence”。因此，SCL 的接口连接与 LAD 块的接口连接相同。

后处理

在自动化框架中，作业控制和所提供的顺控的数据结构允许包含额外的信息，这些信息不是由序列块直接处理的。因此，自动化框架的设备模块包含两个额外的程序段，用于处理与作业和序列执行相关状态信息的后处理。这些程序在序列的状态块下方调用。下面的图片展示了提供额外信息的辅助功能的调用。

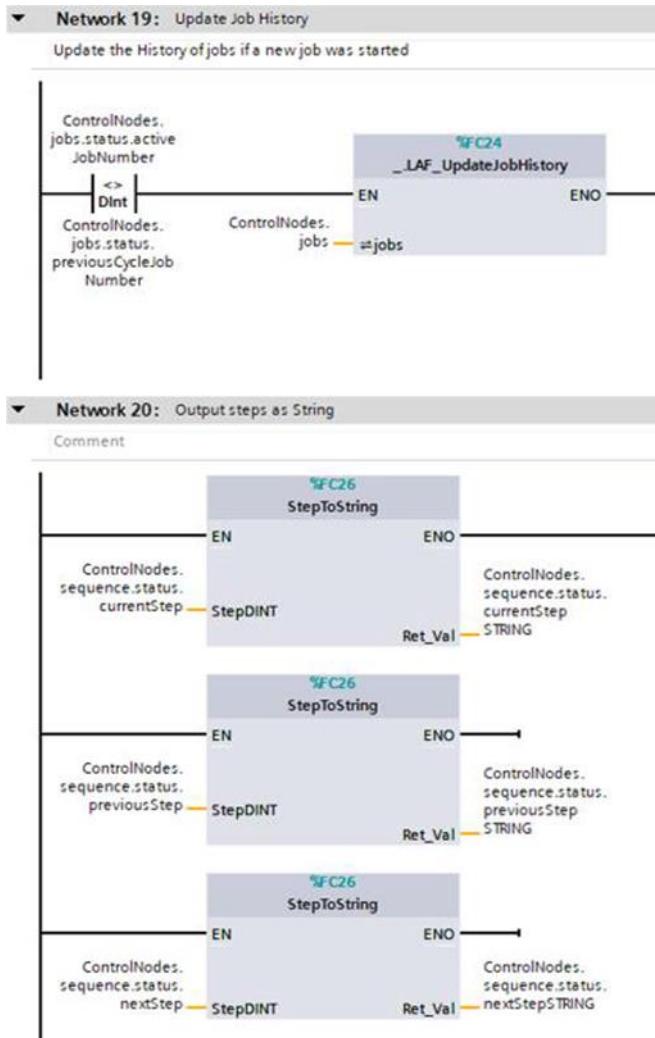


图 10-16 调用函数以获取作业和序列的其他信息

块“LAF_UpdateJobHistory”用于将已执行的作业插入到历史记录中。如图所示，如果活动作业编号不等于上一个周期中处于活动状态的作业编号，则将执行该块，这意味着最后一个作业已完成。函数块“StepToString”只是将步骤编号转换为字符串。因此，当前、上一步和下一步都会调用该块。

10.9. EM 模板/示例

技术说明

模板设备模块是展示主要结构和接口的蓝图 EM，可以作为创建自己的设备模块的基础。在 AF 中，根据整个单元的编程语言、EM 和模块的顺序，提供了四种不同的模板设备模块。

两种 EM 的用例不同之处如下：

- **Em_Template:** 仅包含必要的块和调用。它们是启动自己的设备模块的基础。每种编程语言对应一个模板。
- **Em_Example:** 包含设备模块的基本示例。每个 Em_Example 包含四个 CM 并实现一个序列示例。所有 Em_Examples 都旨在执行相同的用例，以便用户可以轻松比较不同编程语言中的不同实现。

架构

示例和模板 EM 的结构中也采用了 EM 的基本结构（如[接口和块](#)一章中所述）。为了大致了解各个 EM 之间的差异，下表显示了它们的内容：

软件单元	描述
U1_Unit1_SCL	完全在 SCL 中编程
U1Em0_General_SCL	完全在 SCL 中编程
U1Em0_Template_SCL	完全在 SCL 中编程
U1Em1Example_SCL	完全在 SCL 中编程
U2_Unit2	在 LAD 中编程
U2Em0_General	在 LAD 中编程
U2Em1_Template_LAD	用 LAD 编程的通用逻辑 + 在 LAD 中编程的顺控
U2Em2_Template_GRAPH	用 LAD 编程的通用逻辑, 用 GRAPH 编程的顺控
U2Em3_Template_SCL	用 LAD 编程的通用逻辑, 用 SCL 编程的顺控
U2Em4_Example_LAD	用 LAD 编程的通用逻辑 + 在 LAD 中编程的顺控
U2Em4_Example_GRAPH	用 LAD 编程的通用逻辑, 用 GRAPH 编程的顺控
U2Em4_Example_SCL	用 LAD 编程的通用逻辑, 用 SCL 编程的顺控

表 10-1 关于模板和示例 EM 的概述

客户可以使用 EM 模板来启动自己的单个设备模块，同时保持项目的结构。除了示例之外，该模板还提供了一个与用例密切相关的程序，其中包含具有代表性的 CM 和序列。函数块“CallEquipment”的增强程序如下图所示。

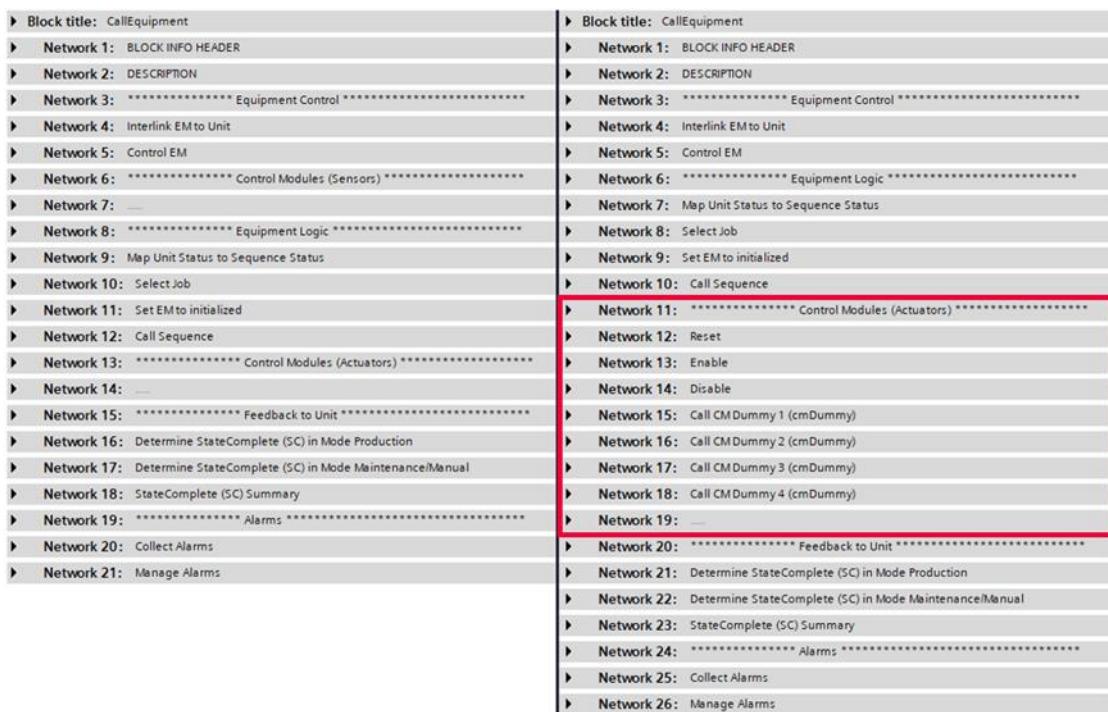


图 10-17 模板和示例 EM 的比较

FB “Sequence” 中使用虚拟 CM 来演示如何构建序列。在序列块中，对各个作业的步骤以及各个步骤的条件进行编程。每个作业都分为不同的步骤。一个步骤代表过程中的一个动作，例如注水或移动电机。在此示例中，作业和步骤的用途更加通用。可以在单元 3 的 EM 中找到完全编程的序列（[EM 演示实例 – 料斗](#)到[EM 演示实例 – 高性能运动](#)）。下图显示了单个步骤的通用结构。

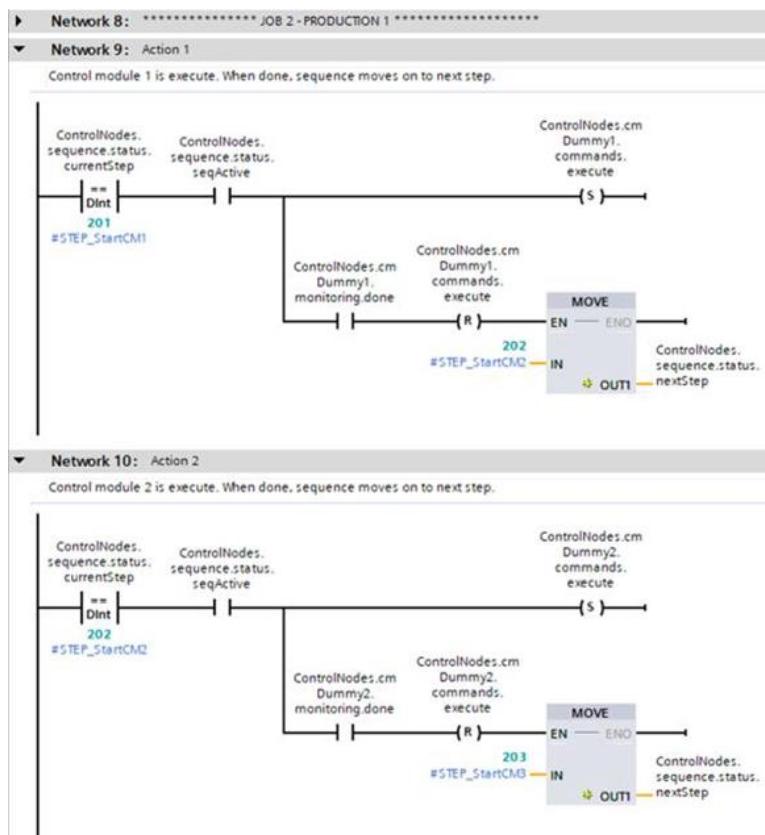


图 10-18 具有步骤结构 (Unit2Em4) 的通用编程示例

接口功能说明

DB “Config” 提供了一个中心点，可以在其中创建参考指示符、轴和 CM 组态。此外，该 EM 的接口分发到 DB “ControlNodes” 以及 DB “HmiInterface”。DB “ControlNodes” 保存 EM 本身的运行系统数据、序列以及用于控制和监控 CM 的作业数据和变量。在 EM 级别，所有进出 HMI 的命令都通过 DB “HmiInterface” 传递。它包含的对象与存储在 DB “ControlNodes” 中的对象相同，但为这些对象提供 HMI 接口值。

10.10.EM General

技术说明

设备模块包含整个单元所需的常规功能、检查和逻辑。它可以包括常规发布和系统级功能，如气压和功率处理以及对事件的响应。

架构

EM General 的架构源自常规结构，[常规结构](#)已经解释过。因为这个 EM 中没有顺序过程，所以没有序列 FB。“CallEquipment”的架构也遵循通常呈现的架构。在此函数块中，功能的实现可以从以下几个方面看到：

- **互连：**EM General 始终与该单元保持联系。
- **常规发布：**包括评估常规功能的信息，如断路器或电源的反馈。
- **控制模块：**不同的 CM（如堆叠灯）是常规功能的一部分。

接口功能说明

此 EM 的接口被分发到 DB “ControlNodes” 以及 DB “HmiInterface”。DB “ControlNodes” 提供有关 EM 状态的信号，包括 interlinkMode、EM 的命令和状态信息。还有一些变量表示在常规发布中处理的 CM 和关联硬件的状态。例如，这可能是熔断器或压缩空气的状态。

DB “HmiInterface” 仅包含 EM 信息，但通过 EM 的报警状态和 UDT 进行扩展，通过仿真单元状态实现直接 EM 控制。

10.11.EM 演示实例 – 料斗

设备模块“Hopper”是一个示例，演示了如何实现料斗应用。它没有非常详细地编程所有细节。它演示了如何通过创建技术模块(TM)来实现和处理更复杂的控制模块。

示例实现包含以下类型的控制模块：

- 插入“LBC_AnalogScaleCn”以实现称重传感器示例。

技术模块“TM Aspiration”包含：

- “LBC_AnalogInput”测量水箱的液位。
- “LBC_PT1-Filter”，用于过滤测得的填充液位。
- “LBC_AxisControl_TectPlcCn”控制搅拌机的速度。
- “LBC_ThreeWayActuator”用于实现料斗的出口。
- 添加“LBC_TwoWayActuator”以控制排水阀，该阀打开和关闭以使剩余物料从料斗中排出。

技术说明

通过实现设备模块，将 OMAC 状态模型与 EM 级别分离。为了给 EM 创建模块化行为，设备可以提供的各种功能被划分为单独的作业。作业可以是 EM 的开始，也可以是生产的一个子过程，例如在新的装载和生产周期开始之前排放剩余的物料。在这些示例中，定义了五个作业。可以在下面找到有关各个作业的详细信息。

在启动作业时，料斗被清空。上一批生产可能会有一些残留物。残留物通过排水阀（双向执行器）排出。下一项作业描述了在启动过程之后如何通过三通执行器将水箱充满水。然后使轴运动，阀返回到中立位置。然后将物料从上方手动添加到料斗中。接下来是生产作业，在该作业中，生产计时器将运行。接下来是排料作业。在这项作业中，生产阀（三通执行器）的出口被打开。当料斗为空时，轴停止。现在可以重新启动该过程。

为了在 OMAC 状态模型和 EM 之间建立联系，有必要将单元的状态与 EM 的不同作业关联起来。例如，OMAC 启动状态将启动 EM 的初始化作业。中止状态会将序列转换为空闲步。此外，初始化作业的结束将更新 EM 确认位，以将状态报告回单元级别。

架构

EM Sequencer 架构允许客户将不同的单元变体关联到设备模块。它基于模板设备模块的架构构建，包含 FB “CallEquipment”中的所有功能。函数块在单元状态和序列之间建立连接。还包括其他程序段，用于管理序列模式的选择、作业以及停止序列并将其返回到其初始状态的条件。如果未选择任何作业，步顺控将保持在空闲步。

EM Hopper 包含以下作业：

- 启动

- 由 OMAC 启动状态触发
- 在下一个作业开始之前清空料斗（如果有任何残留物）
- 执行器设置为空档

- 装载

- 由 OMAC 执行状态触发
- 液体灌装开始，混合器开始移动
- 手动灌装正在启动

- 生产

- 在上一个作业完成时触发
- 搅拌机正在移动
- 混合计时器正在运行

- 排料

- 然后在生产作业完成后执行
- 搅拌机正在移动
- 生产阀处于工作位置，以实现清空过程

- 立即停止

- 由 OMAC 停止状态触发
- 轴将停止，执行器设置在空档位置

设备模块的主要流程和各个步在 FB “Sequence” 中实现。

技术模块 (TM) 和控制模块 (CM)

在此 EM 中，使用了技术模块 (TM) 的结构。TM 是一个包含多个控制模块 (CM) 的模块，因此构成了一个自己的模块。然而，对于设备逻辑，它的行为类似于控制模块，因为它可以通过 DB “ControlNodes” 中的命令和监视位进行控制和监视。此结构可实现集中访问。

在本例中，实现了“cmAnalogScale”和技术模块“tmAspiration”。

代表称重传感器的控制模块“cmAnalogScale”直接在 FB “CallEquipment” 中调用。在此示例中，仿真中未使用该模块。它可以在特定用例中添加。

对于更复杂的模块，定义了一个名为“TmAspiration”的技术模块。它包含在上部分列出的功能。LBC 块用于此目的。“TmAspiration”是如何创建用户自定义复杂模块的示例。遵循 ISA-88 标准，可以在一个控制模块中嵌套另一个控制模块。

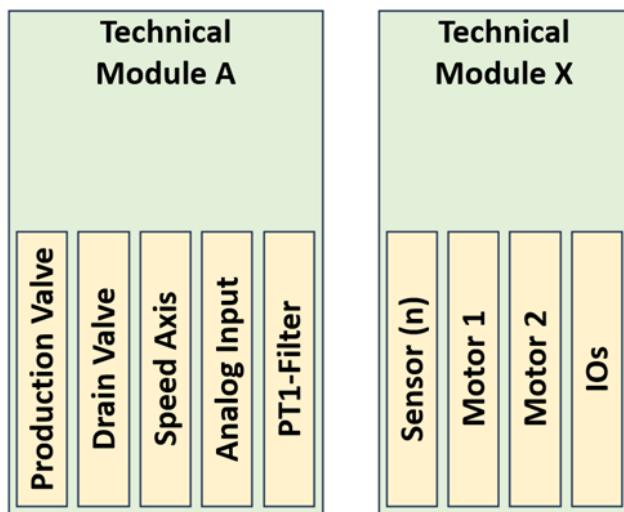


图 10-19 技术模块的概念

接口功能说明

接口 DB 包含一个 EM 控制 UDT，其中包括 EM 的“interlinkMode”命令、状态信息、HMI 部件的仿真单元状态和当前报警状态。此外，还有一个作业控制 UDT 负责管理不同的作业。此 UDT 包括选定的作业编号、当前活动的作业编号，并指示作业管理是处于手动模式还是作业被取消。此外，DB “ControlNodes” 存储已执行作业的历史记录。步顺控的主接口位于 DB “ControlNodes” 和 DB “HmiInterface” 中。HMI 部分包括模式处理、命令和监视信号。在 DB “ControlNodes” 中，该接口包括模式处理、命令和状态，以及先前执行的步的历史记录。

此外，CM 将其接口数据存储在以下两种元素下：DB “ControlNodes” 和 DB “HmiInterface”。

10.12.EM 演示实例 – 传送带

设备模块“传送带站”是一个实现方案，用于演示如何将生产序列与 S7-Graph 块进行集成。主要目的是定义一种模块化的方式，将 Graph Sequences 集成到现有的单元控制模型中。因此，实现的 EM 接口可以连接到不同类型的更高级别的控制变量。

示例实现包含四种不同类型的控制模块。

- “LBC_AxisControl” 函数块，用于控制传送带
- 添加“LBC_TwoWayActuator”以仿真打开和关闭加热站的气缸
- “LBC_DigitalSignal”（3 次）用于接近开关
- “LBC_AnalogInput” 仿真加热站的位置和温度跟踪。

技术说明

设备模块的实现将 OMAC 状态模型与 EM 级别分开。为了实现 EM 的模块化行为，设备可以提供的不同功能被划分为不同的作业。例如，作业可以是 EM 的初始化或生产过程的一部分，例如将生产的对象输送到下一个 EM。OMAC 状态模型和 EM 之间的关系是通过将单元的状态与 EM 的不同作业相关联来建立的。OMAC 启动状态将触发 EM 的初始化作业，或者中止状态会将 S7-Graph 顺控设置为空闲步。初始化作业的完成会设置 EM 确认位以报告给单元级别的状态。

架构

Graph 中 EM Sequencer 的架构允许客户将不同的单元变量连接到提供的设备模块。它基于模板设备模块的架构，具有 FB “CallEquipment” 中的所有不同功能。此外，通过为步顺控提供特殊的控制和监视块，在此函数块中建立了单元状态和 Graph sequencer 之间的连接。EM 传送带站包含以下作业：

- 启动
 - 由 OMAC 启动状态触发
 - 气缸到原点位置，传送带在特定时间内运行以清空剩余的物体
- 执行生产
 - 如果物体到达站的入口传感器，则触发
 - 物体将被运送到站并加热
- 执行删除
 - 如果物体位于站传感器上，则触发
 - 然后在执行生产作业完成后执行
 - 物体被运送到站的出口传感器
- 结束生产
 - 由 OMAC 停止状态触发
 - 轴将被禁用

接口功能说明

em 自身的主接口、作业和步顺控包含在 DB “ControlNodes” 和 DB “HmiInterface” 中。HMI 部分包含命令和监视信号，它们将连接到任何类型的操作点。在 DB ControlNodes 中，该接口由相同的三个元素组成，其中包含先前执行的步的历史记录的附加部分。DB “HmiInterface” 还包含一个用于控制 EM 的特定 UDT，其中可以更改 “interlinkMode” 并监控当前警报状态。在 DB “ControlNodes” 中，其接口主要由相同的元素构成，其中包含先前作业或已执行步骤的附加历史记录，以及针对不同作业和序列的不同启用信号。此外，CM 还在 DB “ControlNodes” 和 DB “HmiInterface” 中保存其接口数据。

ControlNodes [Unit3.Em2]		HmiInterface [Unit3.Em2]	
Name	Data type	Name	Data type
Static		Static	
em	_LAF_typeEmControlNode	em	_LAF_typeEmInterface
commands	_LAF_typeEmCommands	interlinkMode	_LUC_typeInterlinkMode
status	_LAF_typeEmStatus	commands	_LAF_typeEmCommandsInterface
jobs	_LAF_typeJobs	status	_LAF_typeEmStatus
enableManualCtrl	Bool	alarms	_LAF_typeAlarms
commands	_LAF_typeJobsCommands	jobs	_LAF_typeJobsInterface
status	_LAF_typeJobsStatus	commands	_LAF_typeJobsCommands
previousJobs	Array[0..63] of _LAF_typeJobPrevious	status	_LAF_typeJobsStatus
sequence	_LAF_typeSequence	sequence	_LAF_typeSequenceInterface
enableManualCtrl	Bool	commands	_LAF_typeSequenceCommands
enableManualMod...	Bool	monitoring	_LAF_typeSequenceStatus
commands	_LAF_typeSequenceCommands	cmCylinder	_LBC_typeTwoWayActuatorInterface
status	_LAF_typeSequenceStatus	cmTempSensor	_LBC_typeAnalogInputInterface
previousSteps	Array[0..63] of _LAF_typeSequenceStep	cmConveyor	_LBC_typeAxisControlInterface
cmCylinder	_LBC_typeTwoWayActuatorControlNode	cmEntrySwitch	_LBC_typeDigitalSignalInterface
cmTempSensor	_LBC_typeAnalogInputControlNode	cmStationSwitch	_LBC_typeDigitalSignalInterface
cmConveyor	_LBC_typeAxisControlNode	cmExitSwitch	_LBC_typeDigitalSignalInterface
cmEntrySwitch	_LBC_typeDigitalSignalControlNode		
cmStationSwitch	_LBC_typeDigitalSignalControlNode		
cmExitSwitch	_LBC_typeDigitalSignalControlNode		

图 10-20 传送带设备模块的接口

10.13.EM 演示实例 – 高性能运动

设备模块“High Performance Motion”(HPM)是演示如何实现高优先级运动任务的示例。

对于许多机器来说，运动控制程序必须非常快速地执行，这意味着在与轴本身计算相同的周期内执行。这样，可以立即给出新的运动命令，以及对任何运动事件的响应。为了最大限度地利用CPU，将逻辑划分为时间关键型和非时间关键型任务意义重大。这样，可减少CPU负载，并且可以操作更多数量的轴。非运动时间关键型逻辑可以在较慢的周期中执行，其中运动关键型逻辑是在每个运动周期中计算的。非运动时间关键型逻辑还包括基本运动逻辑，如使能、复位或回原点。运动时间关键型逻辑通常包含快速定位动作或对传感器器的响应。

示例实现包含两种不同类型的控制模块，具有以下三个功能：

- “LBC_SinaSpeed”控制正常的速度轴。
- “LBC_AxisControl”定位轴，用于在高性能循环中控制轴。
- “LBC_AxisControl”同步轴，用于控制定位轴的同步轴。

技术说明

高性能运动设备模块的实现遵循与其他演示EM相同的技术实现，将OMAC状态模型与EM级别分离。作业和序列管理块正在实现这种分离。

架构

HPM设备模块提供两个基本作业来实现模块的功能。

- 回原点
 - 由OMAC启动状态触发
 - 定位轴和同步轴回原点
- 生产
 - 由OMAC执行状态触发
 - 启动由“LBC_SinaSpeed”块处理的速度轴
 - 启动高性能程序

如前所述，HPM设备模块包括FB“LBC_AxisControl”的两个实例。这些函数块在设备模块的OB“Main”内的FB“CallEquipment”中正常调用。因此，现有的诊断和HMI功能（例如由LBC块提供）继续在应用中以程序的标准优先级进行处理。

在高优先级运动任务中，直接使用系统指令中的MC_Move命令。这是计算最少、耗时最少的实现方式。“SyncNode”数据块用于标准程序和高性能运动程序之间的数据交换。下图显示了标准程序和高性能运动程序之间的数据流的架构。

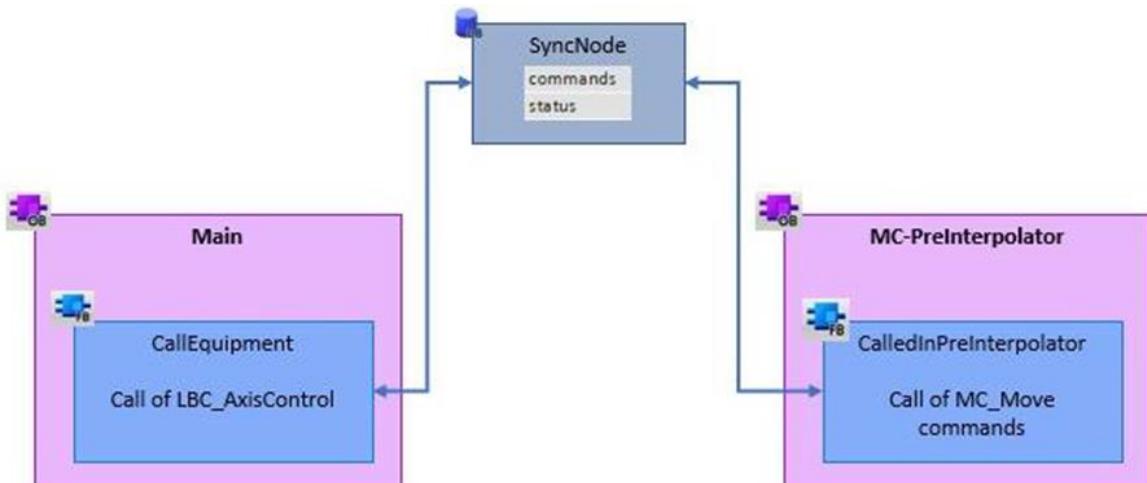


图 10-21 标准周期和运动同步周期的互连

对性能要求不高的操作可以像往常一样在设备模块的顺控中执行，例如启用或复位轴以及对全局操作模式和状态的响应。然后，在顺控的特定步中触发运动循环中快速执行部分的激活。为此，在本例中，使用了一个简易接口。命令“enableHpMotion”被发送到 MotionCycle 中的序列。

确保 IO 的数据一致性也很重要。因此，将运动周期中使用的信号将分配给运动过程映像，并使用 OB “MC_Servo”进行评估。一般来说，建议在运动循环中只执行必要的动作，以避免不必要的负载。

为了实现这种架构并优化 EM 的功能，在运动循环中调用的块存储在高性能运动软件单元中的特定文件夹“CalledInMotionCycle”中。这些块在软件单元运动的 OB “PreInterpolator”中被调用。

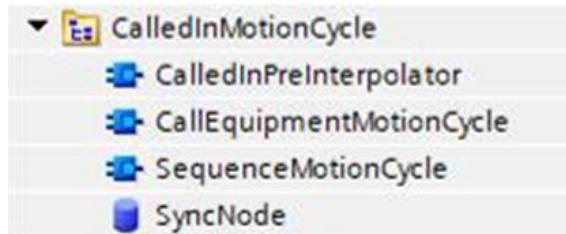


图 10-22 HPM 设备模块的附加块，在运动循环中调用

接口功能说明

HPM 设备模块的接口也遵循 EM 的标准，在 DB “ControlNodes”内实现 PLC 内部信号，在 DB “HmiInterface”内实现 HMI 相关信号。此外，还有特殊的接口 DB “SyncNode”，在架构说明中已经提到过。该接口包括激活高性能运动命令，并提供有关运动任务是否繁忙或是否发生任何错误的状态更新。

10.14. 如何添加新设备模块

添加新设备模块时，必须执行以下步骤。例如，在 U4_Unit4_Dummy 中添加了另一个设备模块。

- 在软件单元 U4_Unit4_Dummy 中，转到 PLC 变量并选择 Unit4 PLC 变量表。在“用户常量”选项卡中，必须执行以下步骤。

Unit4				
	Name	Data type	Value	Comment
1	U4_NO_OF_EMs	Int	1	Highest Value of equipment modules in the unit (Counting starts from 0)
2	U4_EMO	Int	0	Equipment Module 0
3	U4_EM_NEW	Int	1	This is the new EM
4	<Add new>			

图 10-23 Unit4 PLC 变量列表

- 将 U4_NO_OF_EMs 增加 1，以显示单元内最后一个设备模块的最大值。
- 使用附加 EM 的名称添加一个新常量，并分配下一个可能的更高数字。
- 复制最适合需求的 AF 的现有 EM。在这种情况下，U2Em1_Template_LAD 软件单元将被复制，因为需要带有 LAD 序列的新设备模块。右键单击 PLC 项目树中的软件单元文件夹，然后选择粘贴，将其粘贴到项目中。
- 在新的软件单元中，根据需要调整名称和命名空间。因此，右键单击新的软件单元并打开属性。
- 输入新的名称和命名空间后，单击“应用于所有底层元素”，将命名空间应用于软件内的所有对象。

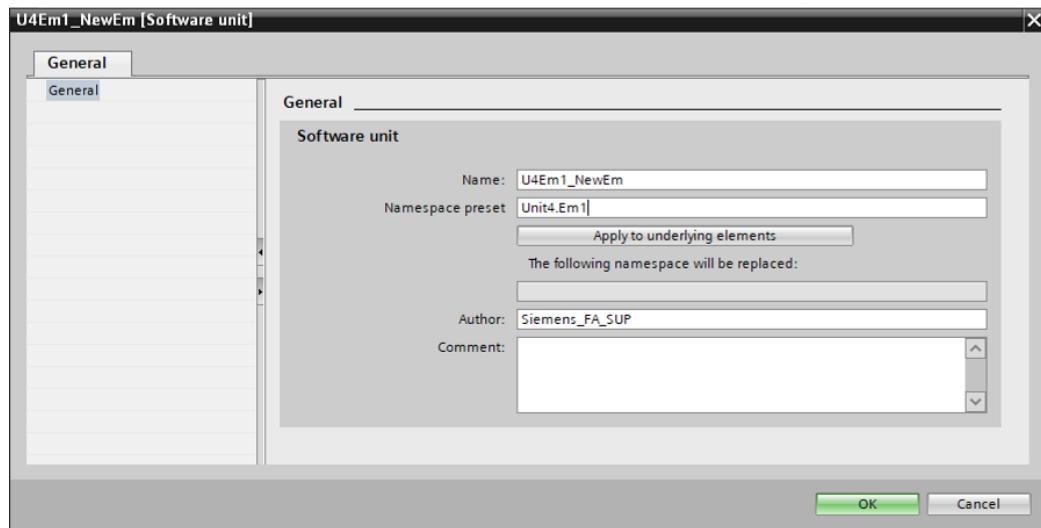


图 10-24 新软件单元的属性

- 通过选择关系菜单并添加新连接，调整与它所分配到的单元的关系。

Relations			
	Selected unit	Relation type	Accessible element
1	▼ U4Em1_NewEm		
2	U4Em1_NewEm	Software unit	ObjectsOfProjectLibraryUsed
3	U4Em1_NewEm	Software unit	U4_Unit4_Dummy
4	<Add new relation>		

图 10-25 对新添加的 EM 关系进行调整

- 打开新 EM 的 OB “Main” 并分配在单元的 PLC 变量中生成的变量。该变量将 EM 分配给单元数据的数组结构。
- 将“CallEquipment”块的接口更改为相应的单元信号。

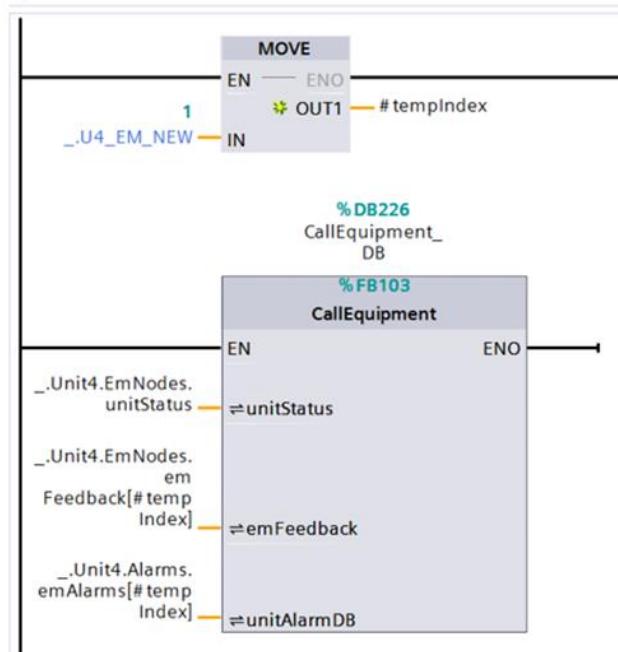


图 10-26 OB Main 调整

- 模板 EM 包含两个作业，可以根据需要调整或添加新作业。作业编号在新 EM 的 PLC 变量中的用户常量中定义。此外，请调整 PLC 变量表的名称以匹配软件单元的名称。

Unit4Em1				
	Name	Data type	Value	Comment
1	U4EM1_JOB1	DInt	1	ID for Job1
2	U4EM1_JOB2	DInt	2	ID for Job2

图 10-27: 用户常量标签页中作业的 PLC 变量定义

- 然后，每个作业都需要一个“LAF_SelectJob”实例，该实例在FB“CallEquipment”的相应区域中被调用。
- 根据[PLC启动和循环调用序列](#)一章中介绍的OB编号约定更改OB“Main”、“Startup”、“ProDiag”的OB编号。

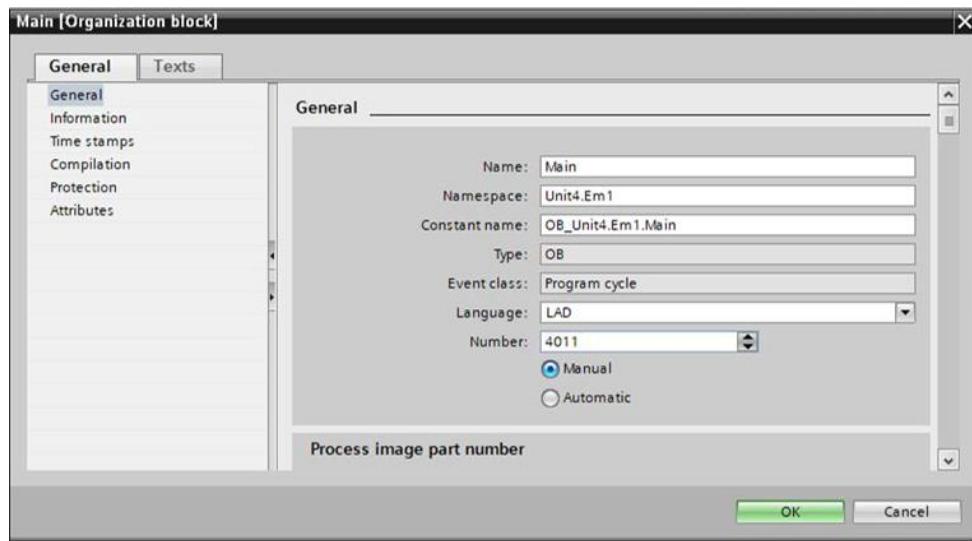


图 10-28 OB Main 的属性

- 通过编译PLC对其余块重新编号，并选择自动解决冲突，如图 10-29。
- 然后，软件单元已准备就绪，可以根据设备模块应完成的流程进行调整。因此，调整序列并添加所需的所有控制模块。
- 为了在EM中获得标准化诊断，可遵循[诊断原理](#)。

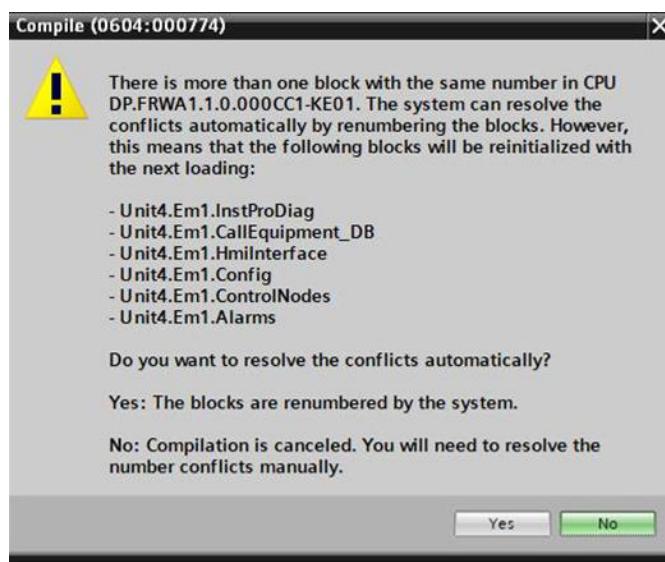


图 10-29 提示解决块编号问题

11. 控制模块

本章重点介绍控制模块 (CM)。它描述了接口结构、如何处理每个模块的组态以及手动和自动控制期间的行为。

控制模块用于控制真实的机器部件或将来自机器的信号转换至设备模块。

11.1. 接口

CM 的接口遵循 PLCoOpen，所有 CM 都至少具有输入“enable”和输出“valid”、“busy”、“error”和“status”。在 FB 输入中，还有执行功能的命令和硬件反馈。在该项目中，使用了 LBC 的控制模块。



有关 LBC 库及其接口的更多信息：

<https://support.industry.siemens.com/cs/ww/en/view/109792175>。

为了便于使用，AF 使用了 LBC 的特殊控制节点版本。控制节点 FB 通过将命令输入移动到命令 UDT 并将输出移动到监视 UDT 来简化 FB 接口。命令和监视 UDT 都在控制节点 UDT 内。此外，还有命令组态 UDT，用于设置命令的值。这简化了模块的使用，并且 CM 的中央数据点可用于处理设备模块内的信号。

11.2. 组态

LBC 中的控制模块有一个组态 UDT，该 UDT 至少用于组态 HMI 和监视的控制模块名称，以及是否应禁用监控报警。

如接口和块一章中所示，设备模块中的控制模块组态存储在“Config”DB 中。建议有两种组态 DB 的方法。

- 第一种是在 DB 中设置起始值，如下图所示：

Config [Unit3.Em2]		
Name	Data type	Start value
1 ▾ Static		
2 ▾ ▾ em	_LAF_typeEmConfig	
3 ▾ ▾ axis	Array[1..._AMOUNT...]	
4 ▾ ▾ cmCylinder	_LBC_typeTwoWay...	
5 ▾ ▾ cmTempSensor	_LBC_typeAnalog...	
6 ▾ ▾ referenceDesignator	WSTRING[25]	WSTRING# "Temperature sensor"
7 ▾ ▾ physicalUnit	String[10]	'Celsius'
8 ▾ ▾ isUnipolarSignal	Bool	TRUE
9 ▾ ▾ default	LReal	0.0
10 ▾ ▾ limitHigh2	LReal	100.0
11 ▾ ▾ limitHigh1	LReal	90.0
12 ▾ ▾ limitLow1	LReal	-10.0
13 ▾ ▾ limitLow2	LReal	-20.0
14 ▾ ▾ processValueMax	LReal	100.0
15 ▾ ▾ processValueMin	LReal	0.0
16 ▾ ▾ scaleAnalogUpPoint	LReal	27648.0
17 ▾ ▾ scaleAnalogLowPoint	LReal	0.0
18 ▾ ▾ scaleProcessUpPoint	LReal	100.0
19 ▾ ▾ scaleProcessLowPoint	LReal	0.0
20 ▾ ▾ disableAlarms	Bool	false

图 11-1 组态示例：直接在 DB 中

- 第二个选项是在启动 OB 期间设置值，如下图所示：

```

1 // Assign axes
2 Config.axis[1] := _U3Em2_SpeedAxis;
3
4 ControlNodes.cmConveyor.commandConfiguration.jog.velocity := 10.0;
5 ControlNodes.cmConveyor.commandConfiguration.moveVelocity.velocity := 100.0;
6

```

图 11-2 组态示例：启动 OB

使用 Startup OB 的优点是，可以根据其他值动态设置值。

11.3. 操作模式

控制模块具有手动和自动控制模式。在这里，手动控制由 PLC 启用，并由用户在 HMI 中控制。只能在单元的手动模式下启用手动控制。以 LBC 为例，输入“enableManualCmds”仅与“#statUnitStatus.mode.manual”一起启用，如下图所示：

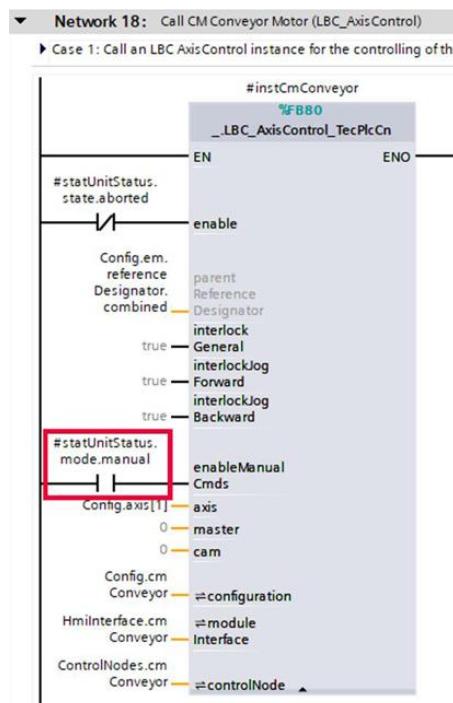


图 11-3 启用 LBC 块的手动命令

当激活 LBC 块的手动模式时，两个接口都可以控制块的内部功能。“moduleInterface”可以通过 HMI 进行控制，PLC 内部命令可用于通过“controlNode”InOut 引脚控制块。

如果未设置“enableManualCommands”输入，则 CM 正在执行自动控制。因此，它只能通过“controlNode”输入进行控制。激活自动模式时，“moduleInterface”将不会被处理。

12. 安全

AF 为安全程序引入了模块化概念。它演示了如何集成安全程序，如何将机器划分为不同的安全区域，以及如何构建安全程序。

注 有关西门子安全编程最佳实践的更多信息，请参见安全编程指南：
<https://support.industry.siemens.com/cs/ww/en/view/109750255>

12.1. 基本信息

功能安全

实施功能安全是一个强制过程，它从可能存在危险的机器开始。为了获得安全的机器，必须了解风险并充分降低风险。

安全系统执行安全功能，由以下子系统组成：

子系统 T1：检测（位置开关、光幕等）

子系统 T2：评估（故障安全 CPU、安全继电器等）

子系统 T3：响应（接触器、变频器等）

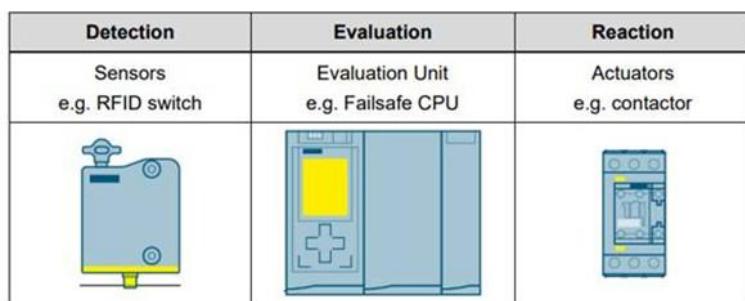


图 12-1 安全系统由多个子系统组成

TIA Safety 与标准 TIA 一样，由两个主要部分组成：参数化和编程。为了实现安全功能，必须正确组态这两个部分，以达到必要且成熟的安全水平。

安全管理编辑器：

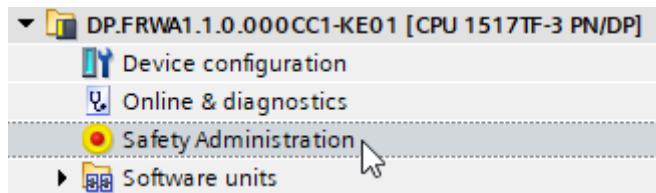


图 12-2 安全管理

安全程序的所有主要安全设置都可以在安全管理中设置。有关详细信息，请参见标准的 TIA Portal 安全文档。

对于 AF，除了以下调整外，将使用默认设置：

- 激活“F-change history”。此设置允许跟踪哪个用户更改或下载了最后更改。出于安全原因，这不能用于检测安全程序中的更改，仅用于编程目的。

- 建议在调试阶段结束时启用“从 F-CPU 一致上传”。这使程序员能够将完整的 PLC 程序从 CPU 加载到空的 TIA Portal 项目中。不建议在调试阶段激活该设置，因为由于必须将数据加载到 PLC 会增加数据，每次下载时间都会增加。

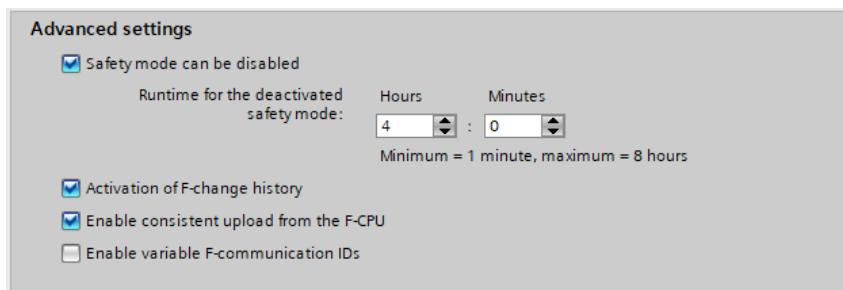


图 12-3 TIA Portal 中的安全程序设置

此外，为防止对安全程序进行不必要的更改，请确保创建密码。

PROFIsafe 地址

PROFIsafe 地址必须是唯一的。在调试期间，必须将 PROFIsafe 地址分配给实际硬件。

注

有关地址类型等的更多信息，请参见西门子安全文档和此应用示例：用于网络级和 CPU 级的唯一的 PROFIsafe 地址的组态示例。

<https://support.industry.siemens.com/cs/ww/en/view/109740240>

硬件 - 通道设置

与标准硬件一样，通道设置可用于直接设置输入和输出信号。可以为输入设置传感器评估、差异行为、差异时间或传感器电源等参数。这些设置在很大程度上取决于应用。必须独立评估每个应用和每个安全功能的必要和正确的安全设置。应在所有项目和模块中禁用未使用的通道。

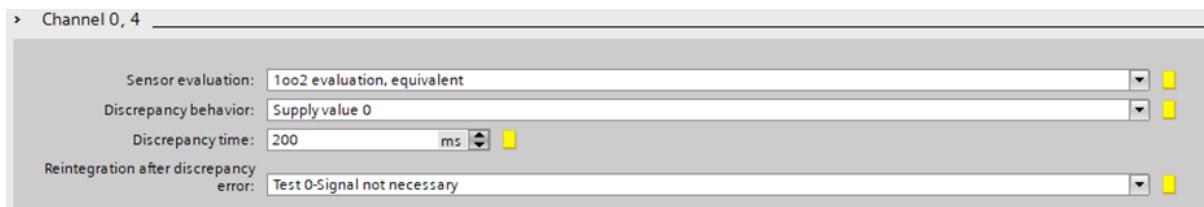


图 12-4 TIA Portal 中的安全通道设置示例

安全编程 – FB Main_Safety_RTG1

函数块 Main_Safety_RTG1 是您自己编程的安全程序的第一个 F 块。该块调用所有逻辑和底层附加功能和功能块。FB “Main_Safety_RTG1”由分配给 F 运行组的 F-OB 调用。此设置可以在安全管理 – F 运行组中完成。

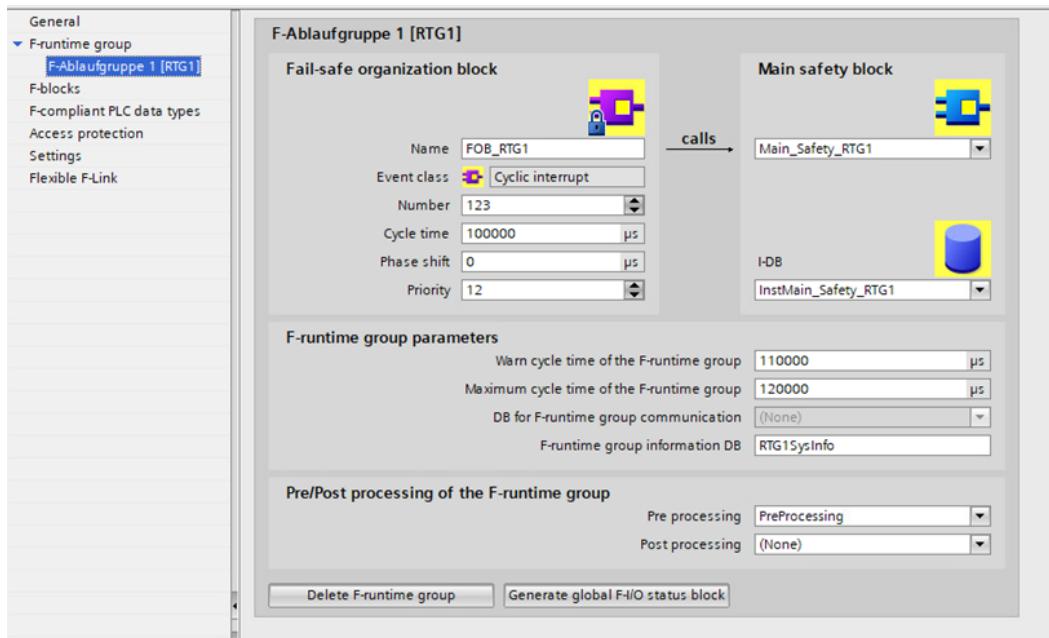


图 12-5 F 运行组的设置示例

在 FB “Main_Safety_RTG1” 中，必须遵循以下块和逻辑调用顺序：

- 从其他 CPU 接收块（F-CPU 到 F-CPU 通信）
- F 模块/F 通道的错误确认/恢复
- 传感器评估块
- 操作模式评估
- 逻辑运算、计算、评估等
- 用于安全执行器的控制块
- 向其他 CPU 发送块（F-CPU 到 F-CPU 通信）

安全功能

对于具有标准化安全功能的简单解决方案，建议使用库。在此示例中，使用了“LSafe”和“LDrvSafe”库。可以在“ObjectsOfProjectLibraryUsed”文件夹中的 Safety 单元中找到 Safety 库。

LSafe 使用基本的安全应用，例如急停或 f 门，以及驱动器或其他执行器的组合。

注

进一步的应用示例：带防护锁的 RFID 安全开关 3SE64 和 ET 200SP 的安全门监控，。

<https://support.industry.siemens.com/cs/ww/en/view/109811081>

“LDrvSafe”库可用于通过 PROFIsafe 和 SIMATIC PLC 控制 SINAMIC 安全功能。

注

有关它的更多信息，请参见 SIMATIC Failsafe 库 LDrvSafe，以控制 SINAMICS 驱动器系列的安全集成功能。

<https://support.industry.siemens.com/cs/de/en/view/109485794>

12.2. 安全和 ISA-88

安全区是机器的一部分，受到安全事件的影响。

常见的情况有两种：

- 整个机器有一个安全区。在这种情况下，安全事件（例如急停）将对整个机器和所有安全执行器产生影响。
- 每个 ISA-88 单元都有一个安全区。在这种情况下，一个单元可以进入安全停止，而其他单元则继续执行。安全传感器可能会对一个或多个安全区产生影响。例如，一台机器可能有一个全局急停按钮，可以停止整个机器，而机器某一部分的光幕只会触发相应单元的安全事件。

AF 演示了一种如何为每个单元实施单独安全区的方法。

12.3. 标准程序和安全程序之间的数据交换

不允许在标准程序和安全程序之间直接交换数据。

在工程组态过程中，对安全程序读取或写入的标准块的更改将导致 F 程序失去其一致性。必须重新编译安全程序，并且只能在 CPU 转入 STOP 模式时下载到 CPU。

在操作过程中，在执行安全运行组期间对安全相关数据进行更改将导致 CPU 停止。

因此，必须严格区分标准程序和安全程序。为了进行标准用户程序和安全程序之间的数据交换，定义了特殊的数据块，其中存储了要交换的数据。此操作允许将标准用户程序和安全程序的块解耦。标准用户程序中的更改不会影响安全程序（反之亦然），因为这些数据块不会被修改。

以下为数据流示意图：

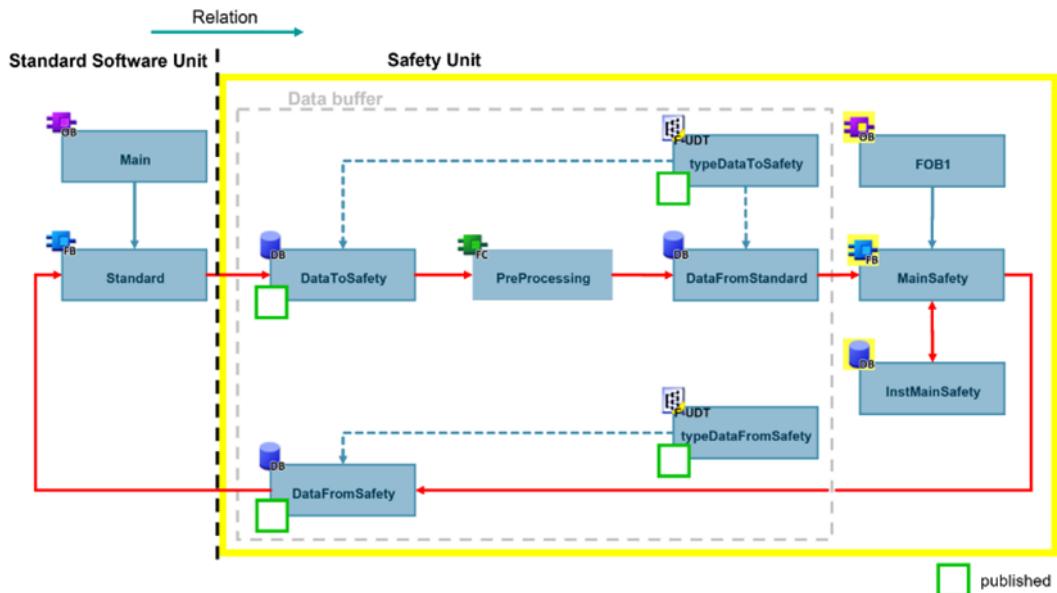


图 12-6 标准用户程序和安全程序之间的数据交换

为此，定义了以下数据块：

DataToSafety

此块包括从标准用户程序发送到安全程序的信息，例如确认命令。

在 AF 中，从标准用户程序发送到安全程序的数据包含以下信息：

DataToSafety [Safety]					
Name	Data type	Start value	Retain	Comment	
1 Static					
2 ackSafetyFromHmi	Int	0		Acknowledge variable from operator control ...	
3 ackSafetyGlobal	Bool	false		Acknowledge command to ack all safety	
4 ackFromUnit1	Bool	false		Acknowledge command comming from Unit 1	
5 ackFromUnit2	Bool	false		Acknowledge command comming from Unit 2	
6 ackFromUnit3	Bool	false		Acknowledge command comming from Unit 3	
7 ackFromUnit4	Bool	false		Acknowledge command comming from Unit 4	

图 12-7 数据块 DataToSafety 的内容

AckSafetyFromHmi 支持来自操作员控制和监控系统的故障安全确认。整数变量用于实现多步确认。详情请见下文。在 AF 中，它与 HMI 的 ACK 按钮关联。

AckSafetyGlobal 提供了给出全局确认命令的可能性。例如，机械硬连线 reset 按钮可以触发该位。

此外，如果安全程序包含硬连线复位按钮，则每个单元都可以向安全程序发送确认命令。在 FB “Main_Safety_RTG1”的程序段 2 中，可以配置哪些安全区应由各个命令确认。

DataFromStandard

该块是一个额外的缓冲块，因此在安全执行时，安全程序中使用的数据不会发生变化。如果未使用该块，则在安全执行期间 “DataFromSafety” 中的任何更改都会导致 CPU 停止。FC “PreProcessing” 将数据从数据块 “DataToSafety” 复制到数据块 “DataFromStandard” 中。两个 DB 具有完全相同的结构 “typeDataToSafety” 。FC “PreProcessing” 可以在安全管理中进行组态。

DataFromSafety

此块包含安全程序中与标准用户程序相关的所有信息。这包括状态信息，例如有源光电保护装置（简称 AOPD）、紧急停止按钮和驱动器。数据的进一步处理，例如用于诊断和报告概念的数据，在标准程序内执行，以保持安全程序的精简。在许多情况下，标准程序在安全事件中执行受控停止或降低轴的速度，而安全程序只是安全监视并交互，例如，如果超过设定的速度边界。

从安全程序发送到设备的数据包含以下信息：

DataFromSafety [Safety]							
Name	Data type	Start value	Accessible f...	Writ...	Visible in ...	Supervision	Comment
1 Static							
2 safetyReleasedAllSafetyZones	Bool	false	✓		✓		All safety zones are released
3 ackRequiredAnySafetyZone	Bool	false	✓		✓		At least one safety zone requires an acknowledge
4 ackOP_changeRequired	Bool	false	✓		✓		ACK_OP is waiting for value change
5 devicesSummary	typeDevicesStatusSummary		✓		✓		General safety status of devices
6 eStopButtonPressed	Bool	false	✓		✓		Global -E-stop button EO is pressed
7 eStopButton1pressed	Bool	false	✓		✓		Zone 1 -E-stop button E1 is pressed
8 eStopButton2pressed	Bool	false	✓		✓		Zone 2 -E-stop button E2 is pressed
9 eStopButton3pressed	Bool	false	✓		✓		Zone 3 -E-stop button E3 is pressed
10 eStopButton4pressed	Bool	false	✓		✓		Zone 4 -E-stop button E4 is pressed
11 iDoor1Open	Bool	false	✓		✓		Zone 1 -F-door is open
12 iDoor2Open	Bool	false	✓		✓		Zone 2 -F-door is open
13 iDoor3Open	Bool	false	✓		✓		Zone 3 -F-door is open
14 iDoor4Open	Bool	false	✓		✓		Zone 4 -F-door is open
15 AOPD1interrupted	Bool	false	✓		✓		Zone 1 -Light curtain is interrupted
16 AOPD2interrupted	Bool	false	✓		✓		Zone 2 -Light curtain is interrupted
17 AOPD3interrupted	Bool	false	✓		✓		Zone 3 -Light curtain is interrupted
18 AOPD4interrupted	Bool	false	✓		✓		Zone 4 -Light curtain is interrupted
19 eStopZone1	typeEstopFunction		✓		✓		Safety status of stop button
20 eStopZone2	typeEstopFunction		✓		✓		Safety status of estop button
21 eStopZone4	typeEstopFunction		✓		✓		Safety status of estop button
22 eStopZone3	typeEstopFunction		✓		✓		Safety status of estop button
23 iDoorZone1	typeDoorFunction		✓		✓		Safety status of door
24 iDoorZone2	typeDoorFunction		✓		✓		Safety status of door
25 iDoorZone3	typeDoorFunction		✓		✓		Safety status of door
26 iDoorZone4	typeDoorFunction		✓		✓		Safety status of door
27 lightCurtainZone1	typeAOPD_Function		✓		✓		Safety status of light curtain
28 lightCurtainZone2	typeAOPD_Function		✓		✓		Safety status of light curtain
29 lightCurtainZone3	typeAOPD_Function		✓		✓		Safety status of light curtain
30 lightCurtainZone4	typeAOPD_Function		✓		✓		Safety status of light curtain
31 statusZone1	typeSafetyZoneStatus		✓		✓		Status of safety zone
32 statusZone2	typeSafetyZoneStatus		✓		✓		Status of safety zone
33 statusZone3	typeSafetyZoneStatus		✓		✓		Status of safety zone
34 statusZone4	typeSafetyZoneStatus		✓		✓		Status of safety zone
35 activeSTO_Zone3	Bool	false	✓		✓		Zone 3 -STD active
36 drive1	typeSafetyDrive		✓		✓		Safety status of drive
37 drive2	typeSafetyDrive		✓		✓		Safety status of drive
38 drive3	typeSafetyDrive		✓		✓		Safety status of drive
39 drive4	typeSafetyDrive		✓		✓		Safety status of drive
40 drive5	typeSafetyDrive		✓		✓		Safety status of drive
41 drive6	typeSafetyDrive		✓		✓		Safety status of drive

图 12-8 数据块 DataFromSafety 的内容

要从安全单元外部访问数据块“DataToSafety”和“DataFromSafety”，必须发布两个数据块和相应的 F-UDT，包括所有从属 F-UDT，如图 12-6 所示。在访问标准软件单元中，必须创建与安全软件单元的关系。这在下面的图 12-9 中作为 Unit 3 的示例显示。

Relations			
	Selected unit	Relation type	Accessible element
1	U3_Demo	... ↳ Software unit	ObjectsOfProjectLibraryUsed
2	U3_Demo	... ↳ Software unit	Global
3	U3_Demo	... ↳ Software unit	Safety
4	U3_Demo	... ↳ Software unit	

图 12-9 与安全单元的关系

在 AF 中，提供了以下安全信息：

- 概要位，显示是否释放了所有安全区，以及显示是否有任何安全区域需要确认的概要位。
- 每个安全区的详细信息
- 所有安全设备的概要
- 每个安全设备（传感器和执行器）的详细信息

概要位可用于监视整体状态。可以使用详细的位来跟踪任何安全事件的根本原因。可以通过 ProDiag 监控来监视设备的各个状态位，例如，HMI 可以显示哪些安全设备已被触发。

12.4. 安全程序

在 AF 中，安全程序被划分为不同的安全区，这些安全区的结构与一个单元相关，每个安全区都有多个从属安全功能，例如紧急停止或防护门监控。每个安全功能内部都包含安全编程指南推荐的结构。

- 评估（相应的传感器，例如紧急停止、防护门等）
- 逻辑（定义对于不同的输入状态相应的安全执行器应如何响应）
- 响应（控制安全功能的相应执行器）

AF 中的安全程序结构遵循上述通用程序结构：

▼ Block title: Main_Safety_RTG1	
Comment	
▶	Network 1: Safety Communication: Rcv from PLC_2
▶	Network 2: Acknowledgement
▶	Network 3: Sensors
▶	Network 4: ***** Safety Zone 1 - Unit 1 *****
▶	Network 5: Safety Function: Emergency Stop
▶	Network 6: Safety Function: Protective Door
▶	Network 7: Safety Function: Light Curtain
▶	Network 8: Actuators
▶	Network 9: Zone State Summary
▶	Network 10: ***** Safety Zone 2 - Unit 2 *****
▶	Network 11: Safety Function: Emergency Stop
▶	Network 12: Safety Function: Protective Door
▶	Network 13: Safety Function: Light Curtain
▶	Network 14: Actuators
▶	Network 15: Zone State Summary
▶	Network 16: ***** Safety Zone 3 - Unit 3 *****
▶	Network 17: Safety Function: Emergency Stop
▶	Network 18: Safety Function: Protective Door
▶	Network 19: Safety Function: Light Curtain
▶	Network 20: Actuators
▶	Network 21: Zone State Summary
▶	Network 22: ***** Safety Zone 4 - Unit 4 *****
▶	Network 23: Safety Function: Emergency Stop
▶	Network 24: Safety Function: Protective Door
▶	Network 25: Safety Function: Light Curtain
▶	Network 26: Actuators
▶	Network 27: Zone State Summary
▶	Network 28: *****
▶	Network 29: Status Summary - Data from Safety
▶	Network 30: Data Exchange: Data from Safety
▶	Network 31: Safety Communication: Send to PLC_2

图 12-10 自动化框架中 FB “Main_Safety_RTG1”的内容

安全通信：从 PLC 2 收发安全数据

这些程序段通过 PRIOFINET 实现从另一个故障安全 PLC 发送和接收故障安全数据。RCVDP 指令从另一个 F-CPU 接收数据。必须在第一个程序段中调用该指令。SENDDP 指令将数据发送到另一个 F-CPU。必须在最后一个程序段中调用该指令。故障安全通信的说明和组态的详细说明可在 SIMATIC 安全组态和编程手册中找到。

<https://support.industry.siemens.com/cs/www/en/view/54110126/171195072139>

确认

该程序段实现了对安全程序的确认和钝化故障安全模块的重新集成。

所有安全区都由一个全局确认信号确认。每个安全区/功能的确认必须适应真实机器的要求。

- **来自 HMI 的确认**

HMI 和 CPU 之间的通信不安全。传输与安全相关的数据需要采取措施确保安全传输。为了使用 HMI 复位安全功能或确认错误，TIA Portal 提供了“ACK_OP”系统块。此块在此处调用。

- **全局确认和来自单元的确认**

这些位可以在 FB “CallGlobal” 中分别设置。AF 中存在空程序段，但未连接任何内容。如果在全局级别或单元中存在机械式和硬连线的 reset 按钮，则可以连接这些按钮并触发这些位。

- **全局确认与重集成**

调用指令 ACK_GL 以创建一个确认命令，用于在通信错误、F-I/O 错误或通道故障后同时恢复 F 运行组的所有 F-I/O 或 F-I/O 通道。

传感器

传感器程序段将来自输入模块的全局地址与局部变量互连，以便在安全程序中进一步使用。

安全区

一个安全区域对应一个单元，由多个安全功能和一个执行器程序段组成。如有必要，可以将安全区进一步划分为对应于不同设备模块的安全区。

安全功能

安全功能是根据风险评估结果实现的一种安全机制，它集检测、评估和反应子系统于一体。每个安全功能都应与其他安全功能相互独立，以确保系统的可伸缩性和降低测试与验证的工作量。

在安全区 3 中，可以根据 LSafe 库找到预配置的安全功能，例如紧急停止。该安全功能在紧急情况下实现驱动器的安全关闭（STO）。当按下紧急按钮时，执行器通过 STO 关闭。

有关 LSafe 和其他可用应用示例的更多信息，常用安全功能可以在 SIOS 中找到：

<https://support.industry.siemens.com/cs/ww/en/view/109773295>

执行器

由于一个执行器可以对应不同的安全功能，因此不能在安全功能程序段内直接控制执行器。该程序段包含对相关安全区的所有执行器的控制。这可能意味着向驱动器发送 STO 命令或切换接触器。每个安全区都有自己的 FB 执行器。

区域状态汇总

总结每个安全功能的诊断信息，这可能包括安全区的释放情况，以及安全区的任何安全功能或执行器是否需要确认。

状态汇总

进一步总结了每个安全区的所有诊断信息。

数据交换

将汇总的诊断信息移动到数据交换数据库，以便在标准用户程序中进一步使用。

12.5. 演示

在 AF 示例中，预先配置和评估了 4 个紧急停止和 1 个保护门监控功能。在安全区 3 中，一个紧急停止和保护门监控功能将导致驱动器的安全关闭（STO）。

AF 默认安全密码为：Siemens!123

借助仿真工具，可以在虚拟、无风险的环境中验证安全程序，而无需大量硬件，从而获得部署就绪的 PLC 代码并大大降低工程成本。安全单元可以用不同的方法进行模拟：

PLCSIM:

使用 PLCSIM，可以创建 SimTable 来监控和修改故障安全输入，以测试和验证已实现安全功能的基本逻辑。

PLCSIM Advanced 与基于 SIMIT 的外设仿真

借助 PLCSIM Advanced 和仿真软件 SIMIT，PLC 行为以及传感器和执行器的行为都可以使用仿真模型进行仿真。这允许对程序进行验证和验证，并测试设备的物理行为。

有关不同仿真工作流程的详细说明，请参见 [仿真](#) 一章。

13. 运动控制

13.1. 概述

S7-1500 运动控制 (MC) 由位于 Motion 软件单元中的 Motion OB 集中执行。从技术角度来看，工艺对象 (TO) 的运动控制指令应该在包含轴控工艺的设备模块中进行编程。不过，运动指令的调用可以在设备模块的标准周期 “Main” OB 或 “MC_Pre-Interpolator” OB 中进行编程，后者与 “MC_Servo” OB (=运动控制应用程序周期) 在同一快速周期中运行。

MC_Servo/ MC_Interpolator

当第一个 TO 添加到 S7-1500 PLC 时，将自动创建用于处理 TO 的 MC_Interpolator 和 MC_Servo OB。TO 的功能根据应用周期创建自己的执行优先级。这些 OB 受专有技术保护，程序无法编辑。

MC_PreInterpolator

MC_PreInterpolator OB 和其他不受专有技术保护的 Motion-OB 可用于在运动循环中调用运动命令，并且必须手动插入。这些 OB 考虑了有关时间关键事件的特殊要求。仅应在需要时对这些 OB 进行编程，例如，在需要快速反应或即时执行定位时，可立即启动运动指令。MC_Power 和 MC_Reset 等标准运动命令应在设备模块的标准程序周期中调用。否则，PLC 会加载太多不必要的程序，从而降低 PLC 的整体性能。

下图详细说明了运动控制应用周期和不同 Motion-OB 的优先级：

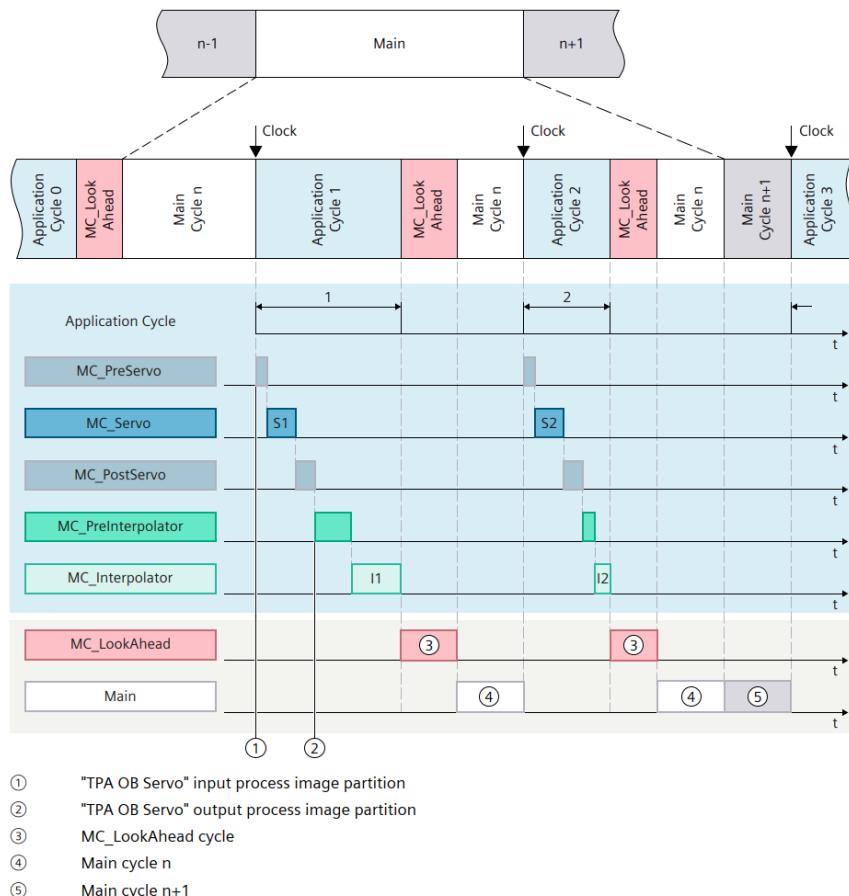


图 13-1 Motion-OB 的运动控制应用周期和调用层级

13.2. Motion 软件单元

Motion 软件单元非常精简。它只包含 Motion-OB 和时间关键程序代码的调用。标准运动命令在相应的设备模块中进行编程和调用。时间关键型运动命令也在相应的设备模块中编程，但在 MC_PreInterpolator OB 中实例化。

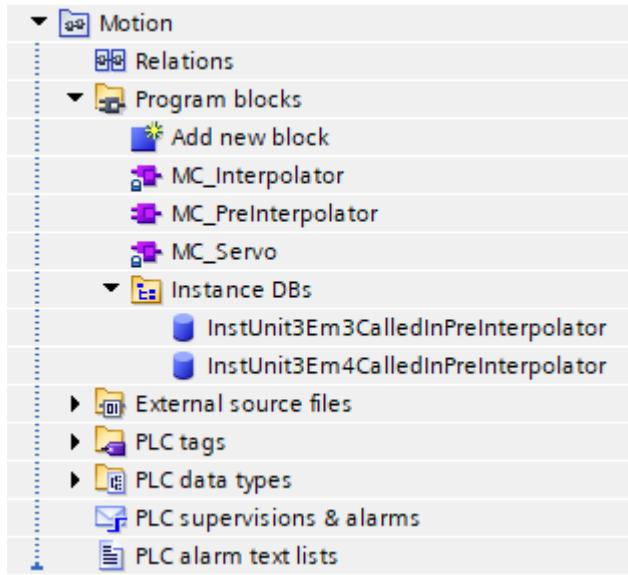


图 13-2 Motion 软件单元的程序块

工艺对象在 PLC 的全局软件单元之外进行组态。要访问 TO 的数据，需要在相应的设备模块软件单元中建立与 TO 的关系。如果需要时间关键型运动应用，则需要从软件单元运动到相应设备模块软件单元的附加关系，以在运动周期中调用时间关键型运动命令。

在 AF 中，时间关键型运动命令在 MC_PreInterpolator OB 中调用，例如，对于单元 3 的设备模块 3：

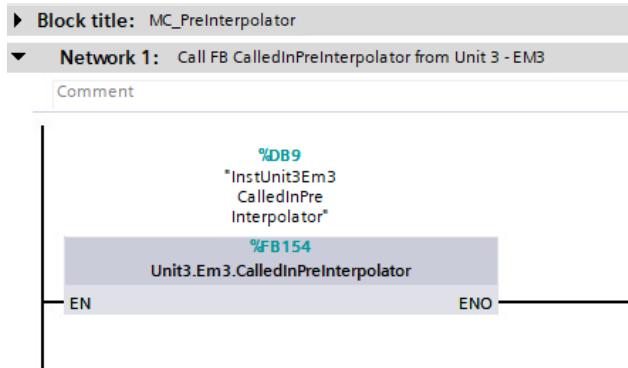


图 13-3 在运动控制应用周期中，从 EM 调用时间关键型运动指令

在 AF 的预组态 TIA 项目中，包含一个演示设备模块“U1Em3_HighPerformanceMotion”，展示了如何将运动控制集成到用户程序中。

注

有关如何实现高性能设备模块的详细指南，请参见章节 [EM 演示实例 – 高性能运动](#)。

13.3. 工艺对象与 Sina-功能块

驱动器可以通过 PLC 内部的工艺对象进行控制，也可以通过 Sina-块独立进行控制。使用工艺对象和 Sina-功能块的区别主要体现在控制器的位置。

使用 TO 时，位置控制器和插补器位于 PLC 中，并向驱动器发送控制命令以及速度设定值。速度和电流控制由驱动器完成。因此，智能系统位于 PLC 内部，它在齿轮传动、凸轮和运动系统方面提供了更强大的功能。

当使用像 SinaPos 这样的 Sina-功能块时，位置控制器和插补器位于驱动器中。PLC 中的 Sina-功能块就像一个接口，用于向驱动器发送控制命令。智能系统位于驱动器内部，仅限于具有速度控制和定位等基本功能的独立轴。

这两个概念是不同的，有其各自优点。下表对这两个概念进行了比较。优势以绿色突出显示。

	工艺对象	Sina -功能块
功能	<ul style="list-style-type: none">速度控制定位传动凸轮运动系统	<ul style="list-style-type: none">速度控制定位（使用 EPOS）
轴数量	取决于 CPU。 <ul style="list-style-type: none">CPU1518TF 支持多达 192 个轴CPU1508S TF 支持多达 310 个轴	轴数仅受程序段限制。
性能	<ul style="list-style-type: none">取决于 CPU 和轴的数量。至少对 CPU 性能有中等影响。	<ul style="list-style-type: none">与轴的数量无关。对 CPU 性能的影响较小。
SINAMICS 驱动器	全部	全部
PROFIdrive 报文	支持范围广泛的报文。	仅限于报文 1（速度控制）和报文 111（通过 EPOS 定位）。
PROFINET IRT	对于伺服驱动器是必需的	不需要
轴组态	<ul style="list-style-type: none">到 TIA Portal 中直观选项丰富	<ul style="list-style-type: none">在 Startdrive 中需要额外的专业知识有限的选项
PLC 编程	<ul style="list-style-type: none">用于 MC 的各种 PLCOopen 块功能性高灵活性高	<ul style="list-style-type: none">仅限于“Sina 块”功能性低灵活性低
可用性	高	中

工艺对象	Sina - 功能块
轴数据可用性	<ul style="list-style-type: none"> 通过 TO 的 DB 直接可用 复杂度低 极少工程组态工作
诊断	<ul style="list-style-type: none"> 支持 CPU 和 Startdrive 跟踪 在 TIA Portal 中进行在线诊断 技术报警可在 CPU 的诊断缓冲区中通过 CPU 显示屏和 Web 服务器获得，也可以在 HMI 的报警视图中找到
仿真	<ul style="list-style-type: none"> TO 集成仿真 无需额外的软件和许可证
开放性/插件	<p>支持</p> <p>支持</p>

表 13-1 TO 和 Sina 块的比较

“高性能运动控制”设备模块 展示了一个 TO_PositioningAxis 定位轴带有 TO_SynchronousAxis 齿轮同步轴，以及一个使用 Sinablock SinaSpeed 的驱动实现速度控制，作为样例。

由于各轴必须同步运动，因此控制所需的智能系统位于 PLC 中，并通过 TO_PositioningAxis 和 TO_SynchronousAxis 实现。

由于速度控制驱动器的运动是独立的，因此可以使用 TO_SpeedAxis 或 SinaSpeed 模块。在高性能运动应用中，为了尽可能减少 CPU 负载，使用 “LBC_AxisControl”中的 SinaSpeed 程序块来控制驱动器。

14. 通信

注

请注意，在没有 TIA Portal V19 Update 3 的情况下，当编译带有已配置 OPC UA 服务器接口的 PLC 时，TIA Portal 会崩溃。更多信息、解决方案和解决方法请参见以下内容：

<https://support.industry.siemens.com/cs/ww/en/view/109971630>

最新日期：18/09/2024

14.1. 机器-MES 通信

在智能制造行业中，各种 IT 应用程序必须与车间自动化（OT – 操作技术）交换数据。这给 IT/OT 集成带来了挑战，例如不同的现场总线协议、要连接的设备数量的可扩展性、交换数据的一致性以及 IT 安全性和可用性方面。

为了应对这些挑战，OPC UA 提供了一套丰富的功能，实现了独立于供应商的集成、独立于 PLC 代码实现的灵活接口、数据建模以及各种数据交换方法，以达到一致性和不可或缺的 IT 安全性。西门子提供了一个信息模型套件作为工具包，用于创建生产级的标准化机器接口，将车间连接到 MES/MOM 等 IT 系统。

西门子信息模型套件提供标准 OPC UA 节点集，其中包含独立于行业的 IT/OT 集成中常用的类型和对象。除了机器信息和状态等通用机器模型外，还可以找到用于交换通用信息的通信方法，以实现生产工作流程的无缝集成。该指南有助于理解如何创建自己的标准化接口，在建模时需要考虑哪些因素，以及如何实施集成用例，如 KPI / OEE 监控或跟踪与追溯。西门子信息模型套件还包含用于将控制器与西门子 MOM 产品直接集成的接口。



西门子信息模型套件提供标准 OPC UA 节点集，其中包含独立于行业的 IT/OT 集成

适用于制造用例的西门子信息模型套件

<https://support.industry.siemens.com/cs/ww/en/view/109814835>

中常用的类型和对象^[6]。

<https://support.industry.siemens.com/cs/ww/en/view/109814835>

此外，AF 还提供了基于西门子信息模型套件，符合 OMAC PackML 数据结构的解释性 OPC UA 服务器接口。示例接口的理念是分享和介绍接口建模的最佳实践，以便更简单、更快速和更具成本效益的实现 IT/OT 集成。

提供的 OPC UA 服务器接口涵盖四个集成用例：

- 机器信息，基于静态数据集，其中包含机器的基本信息和描述。
- 机器状态，显示当前设备状态和模式以及基本运行关键绩效指标。
- 机器监视，对机器中模块或组件的特定状态或过程值进行监视。
- 机器控制，可通过简单的指令远程控制设备状态和模式。

注

使用免费的“Siemens OPC UA Modeling Editor (SiOME)^[3]。”，PLC 工程师可以轻松修改和/或扩展接口，并将其部署到 PLC 的 OPC UA 服务器。

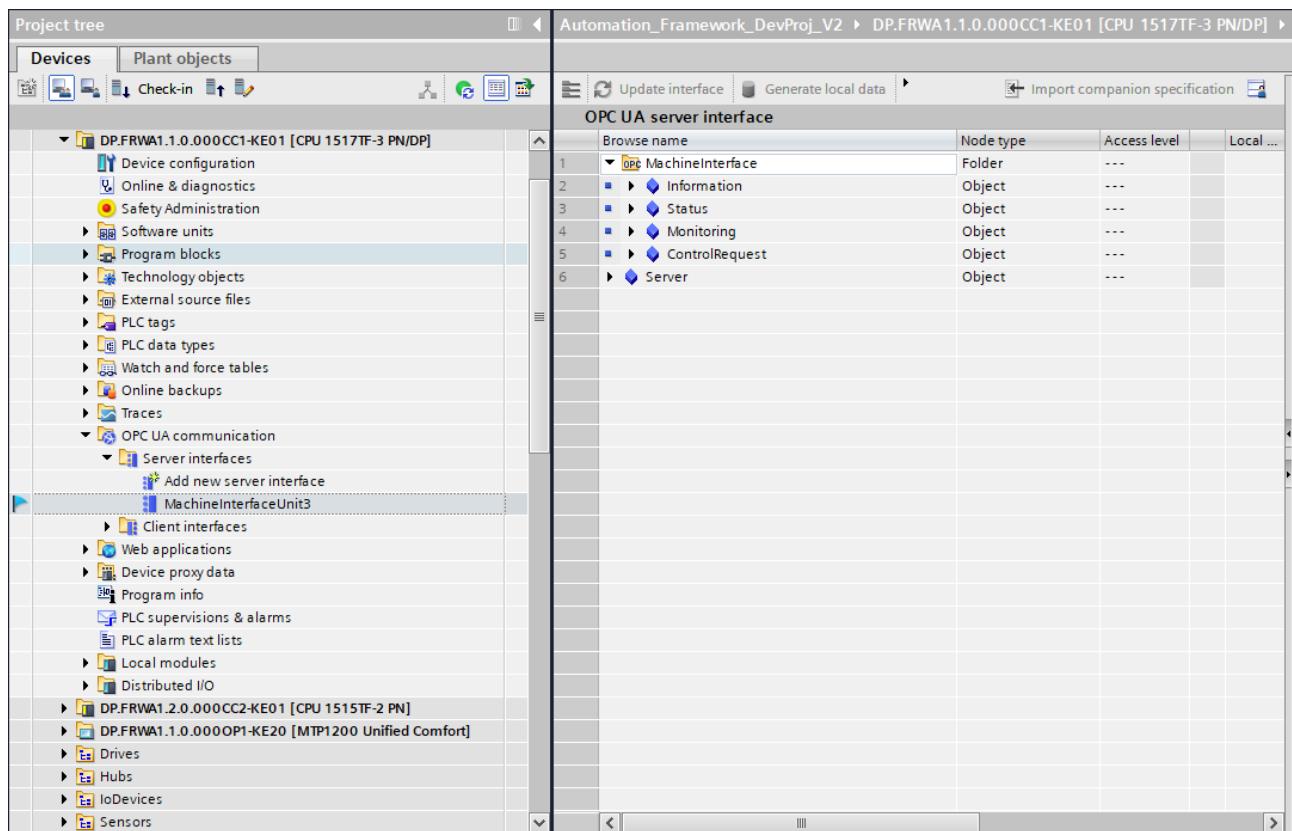


图 14-1 单元级别的 OPC UA 服务器接口

注

OPC UA 服务器接口作为单元 3 的示例实现并映射。它没有经过测试！

此外，单元 3 的软件单元全局内禁用了远程控制接口，这意味着 OPC UA 方法不会影响单元的当前模式和状态。

此外，AF 提供了一个预定义的图标，用于在 HMI 标题栏中可视化机器和 IT 系统之间的连接状态。IT 连接图标显示两种状态：

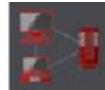
状态	图
已连接（浅灰色）	
未连接（红色）	

表 14-1 IT 连接状态图标

注

PLC 应用程序必须识别其 IT 连接状态并将其映射到以下变量：
“CentralFunctions.Settings.ITConnection”

14.2. 机器-机器通信

PN/PN 耦合器

PN/PN 耦合器允许位于独立网络中的两个 PROFINET 控制器交互数据。PN/PN 耦合器以物理方式将网络隔离开来，并提供 PROFINET 实时 (RT) 通信，并全面支持 PROFIsafe 行规，以实现网络边界上的安全通信。

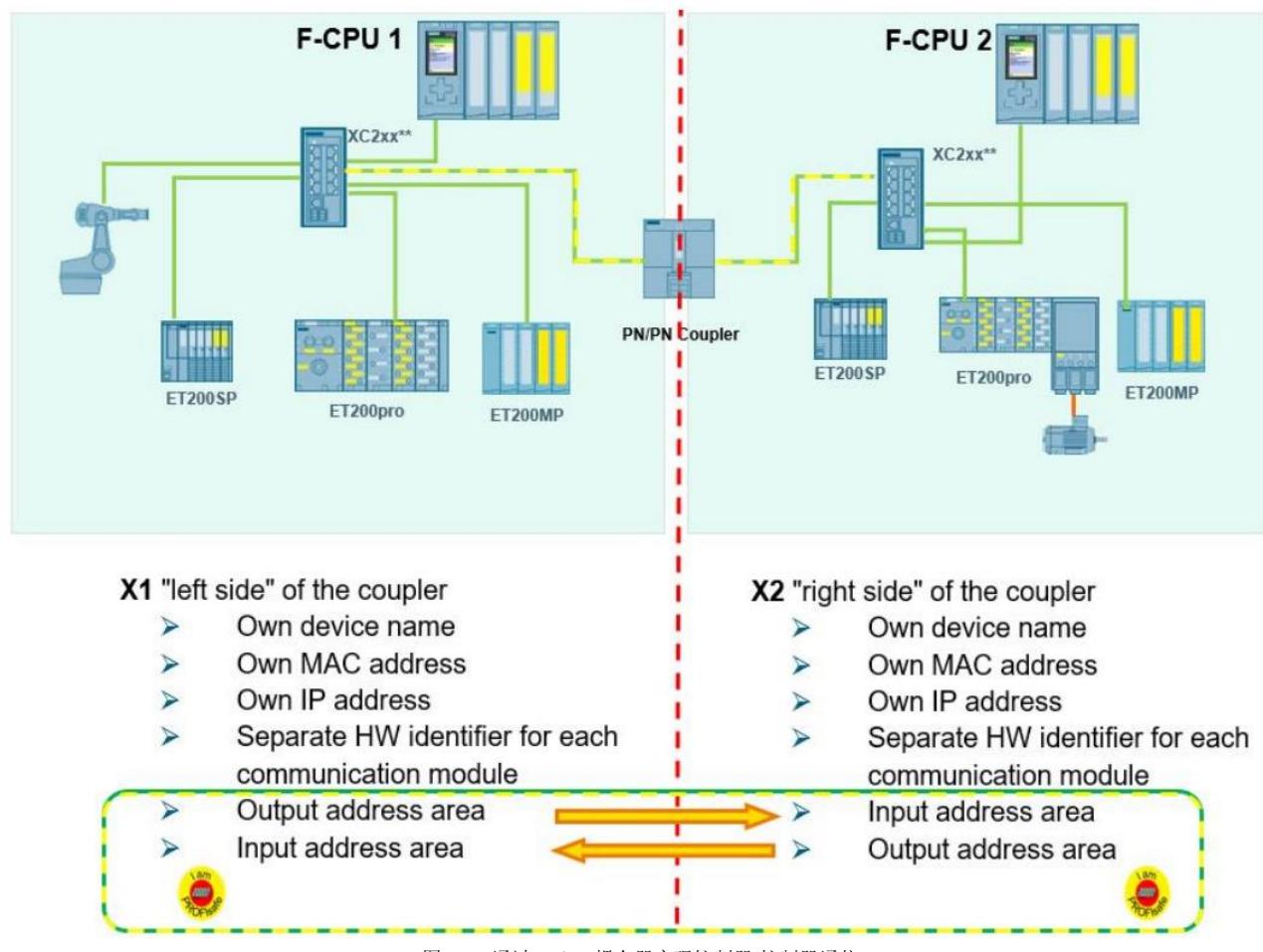


图 14-2 通过 PN/PN 耦合器实现控制器-控制器通信



有关使用 PN/PN 耦合器的详细信息和优缺点，请参见 TIA Portal 的信息系统。

注

AF 提供了一个预先组态的实时通信示例，使用 PN/PN 耦合器在两个故障安全 PLC 之间传输 PROFIsafe 数据。

智能设备

S7-1500 中的智能设备功能允许 IO 控制器同时充当 IO 设备和 IO 控制器。当不需要在机器之间进行网络分离时，或者当更多的 PLC 位于同一子网中并且需要实时交换数据时，这种通信获得最终客户的认可。在这种情况下，PROFINET 控制器之间的通信可以使用智能设备功能来实现。

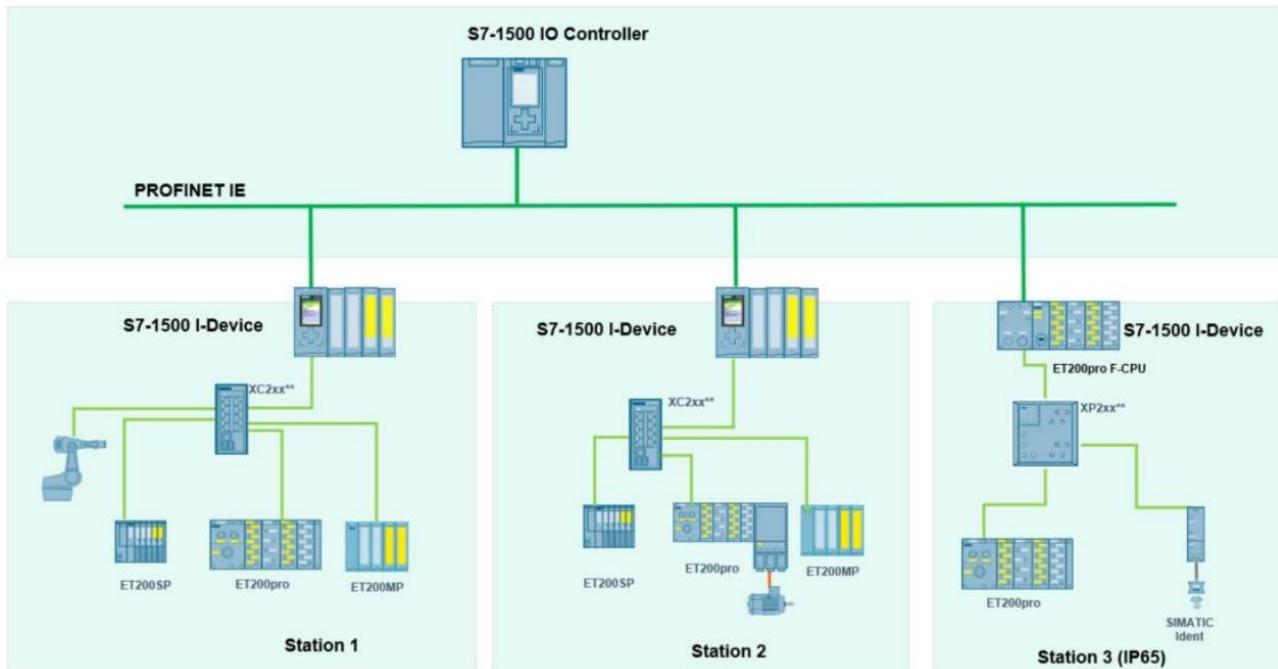


图 14-3 S7-1500 作为智能设备



详细信息以及将 S7-1500 用作智能设备的优缺点，请参见 TIA Portal 的信息系统。

Flexible F-Link

Flexible F-Link 支持跨机器/子网边界的与安全相关的 CPU 到 CPU 通信，并依赖于标准通信协议 (TCP/IP)，提供从一个网络路由到另一个网络（通过第 3 层和聚合级别）的选项。

这不会影响机器安全概念，也不需要对路由器或端口的组态提出任何特殊要求。只需确保网络之间的开放用户通信（例如 TCP 或 UDP）已路由即可。

由于底层通信方法使用标准通信协议，因此 Flexible F-Link 不是一种确定性的通信方法（无实时通信）。特别是对于非常快速的安全反应时间，它并不适合所有应用。如果可以通过 Flexible F-Link 实现与安全相关的通信，则必须根据具体情况进行验证。

此外，Flexible F-Link 不需要任何额外的硬件，并且可以在现有的网络基础设施上实施。

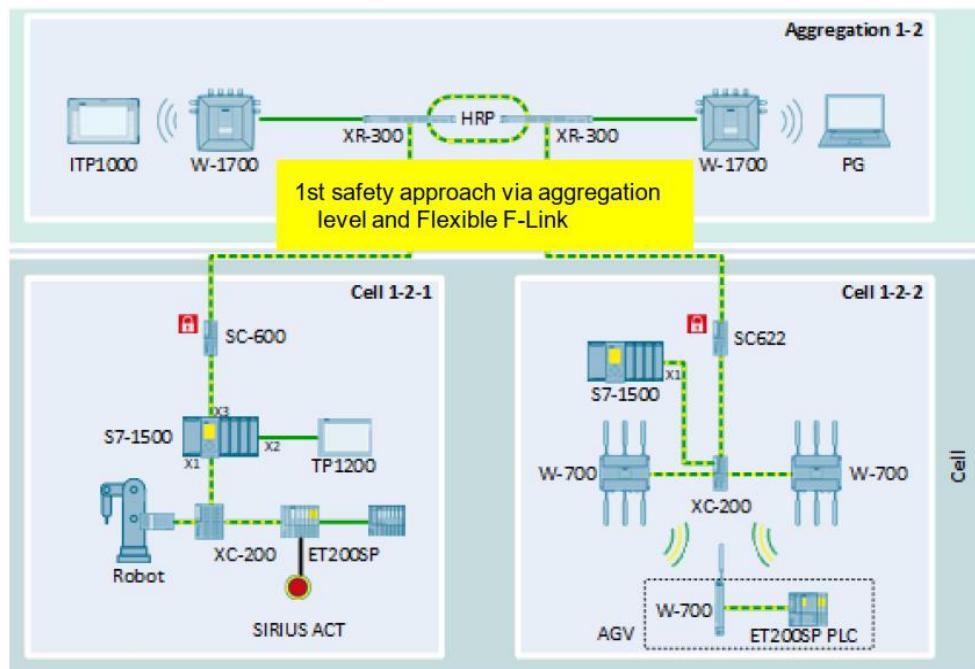


图 14-4 通过 OUC 和“Flexible F-Link”实现 CPU 到 CPU 通信



有关“Flexible F-Link”的详细信息以及“Flexible F-Link”组态的应用示例，请参见 TIA Portal 的信息系统。

注

建议在开发阶段的早期指定安全相关数据，以减少在工程组态和调试期间或之后的临时更改。所选的安全网络架构必须确保 PROFIsafe 地址在整个网络中是唯一的。

此常见问题解答中提供了其他信息：<https://support.industry.siemens.com/cs/de/en/view/109740240>

14.3. 与第三方系统通信

标准化的通信概念 OPC UA 为成功集成设备控制层（如 PLC）与更高级别的工厂自动化（如 IT 系统）（如制造执行系统）提供了基础。

“通过 OPC UA 实现基于消息的一致通信”应用示例提供了一种解决方案，通过基于变量或基于方法的通信，以一致且可靠的方式传输数据。此外，还包括握手机制和缓冲（环形缓冲区）。每个要与机器通信的合作伙伴，只需要按照标准程序与机器交换数据。

此应用解决的要求是：

- 可以处理任意类型的消息内容。
- 可以处理与多个合作伙伴的通信。
- 可以专门处理传入消息以及传出消息。
- 消息的发送顺序与接收消息的顺序相同。
- 缓冲区可防止在通信中断时丢失消息。
- 可以使用的通信机制：OPC UA（注册）读/写、订阅，OPC UA 方法。

这个概念可以通过自定义消息类型来扩展。它通过写入消息缓冲区和从消息缓冲区读取消息来提供可靠、一致和序列化的通信。必须设置单独的报警机制，包括 PLC 和信息系统之间的握手。因此，诊断信息需要统一。发送到信息系统的每条报警消息都有一个 ID、类别和值，以在附加诊断表中指出详细说明。报警消息的握手将由报警管理器组织，该管理器触发被激活的报警并确认报警是否被信息系统确认。



此处提供了详细的文档以及“通过 OPC UA 实现基于消息的一致通信” TIA Portal 应用示例：

<https://support.industry.siemens.com/cs/ww/en/view/109795979>

15. 过程诊断

15.1. 概述

AF 提供了一个整体的诊断概念，可以有效地定位机器应用中的故障和错误。在 TIA Portal 中，“系统诊断”和“过程诊断”被区分开来。

- 系统诊断与硬件密切相关，并在[诊断](#)一章中进行了描述。
- 过程诊断与机器操作有关。PLC 程序中定义的变量受到监控，并为 HMI 或第三方系统创建全文消息。

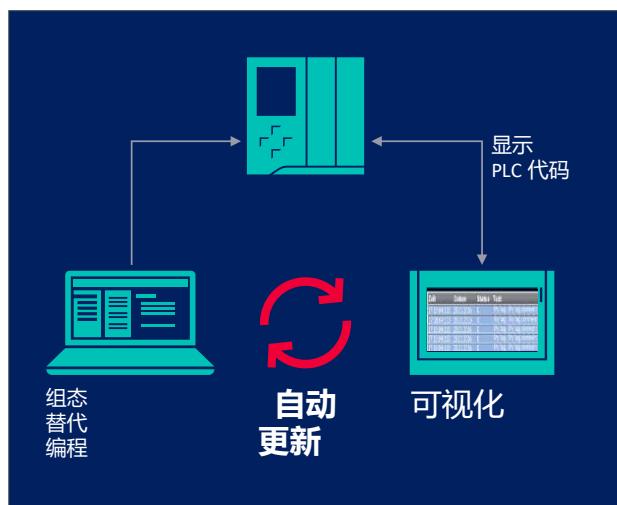


图 15-1 用于过程诊断的 TIA Portal 选件

在 AF 中，ProDiag 选件用于过程诊断。

使用 ProDiag 具有以下优点：

- 一致的报警文本，在 PLC 中自动生成，在运行系统中至多使用三种不同的语言。
- HMI 上无需工程组态。
- 自动生成的收集集，用于管理机器行为。
- 在报警浪涌内，周期时间几乎没有增加。
- HMI 中的标准分析 – 查看根本原因。
- HMI 上的 PLC 代码查看器，用于互锁。
- 只需单击几下即可创建新的监控。

以下章节将介绍 ProDiag 和 AF 实现所需的 TIA Portal 许可证和系统设置。

注

由于其模块化结构，ProDiag 可被删除，并由用户自己的实施方案取代

15.2. 先决条件

本章介绍 ProDiag 的必要设置。满足最终用户要求的最终设置可能会有所不同。



有关 ProDiag 的详细文档可在 TIA Portal 的信息系统中找到，以及用于理解其用法的应用示例：
<https://support.industry.siemens.com/cs/us/en/view/109740151>

15.2.1. PLC 设备设置

激活中央报警

必须在 PLC 设备的属性中激活“PLC 中的中央报警管理”。这将启用 PLC 上的报警服务器。全文消息在 PLC 程序中生成，然后发送到 PLC 报警服务器，PLC 报警服务器将消息推送到所有已连接和可用的报警客户端，例如 TIA Portal 工程组态、PLC 上的显示器、所有连接的 HMI 或 PLC 的 SIMATIC Web 服务器。

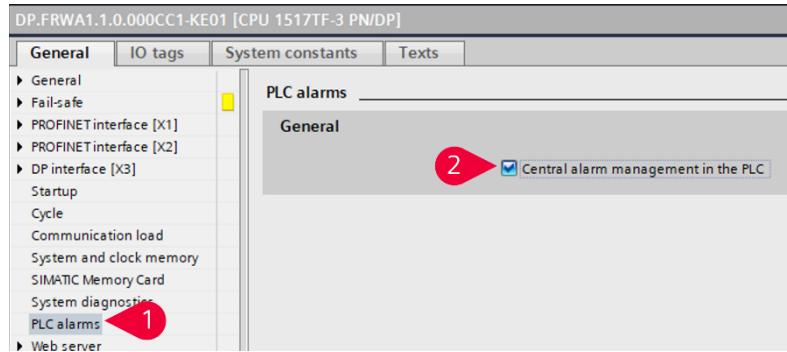


图 15-2 在 PLC 中激活中央报警管理

15.2.2. 多语言支持

在 AF 项目中，默认的 ProDiag 语言设置为英语，但支持多语言。要以正确的语言显示消息，请执行以下操作：

- 在“语言和资源”文件夹中打开项目语言组态。
- 选择要添加到项目中的所有项目语言。

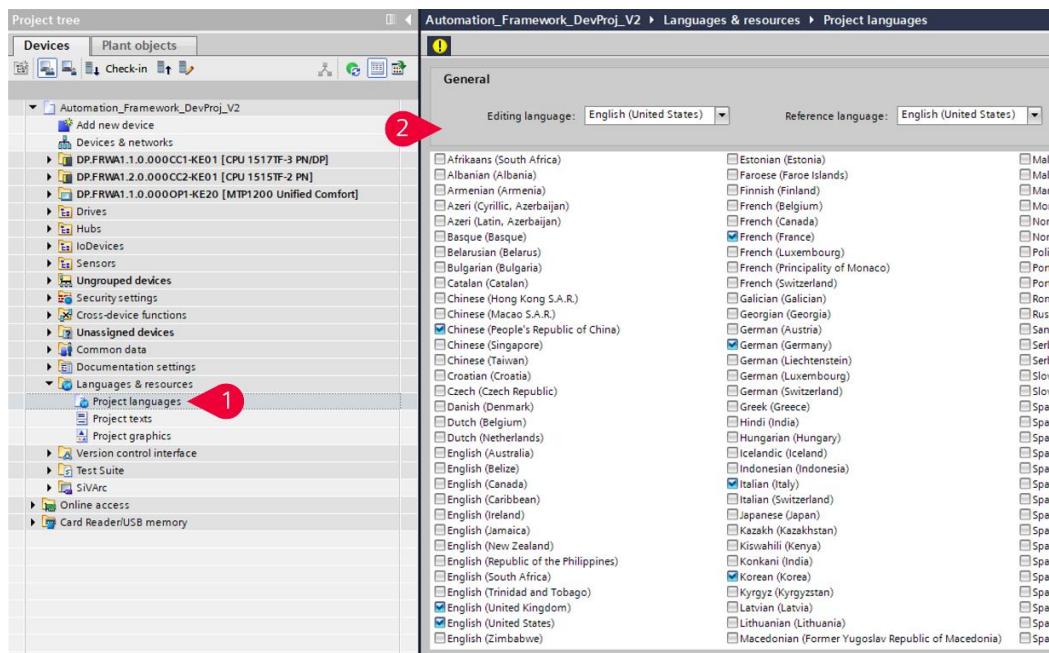


图 15-3 整体项目语言设置

在此组态中激活的所有语言现在都可以在项目文本中进行调整。必须采取的进一步步骤是将项目语言分配给 PLC 及其 Web 服务器的相应系统语言。为此，必须采取以下步骤：

- 在 PLC 的属性中选择多语言支持设置。
- 将应显示的项目语言分配给系统语言。

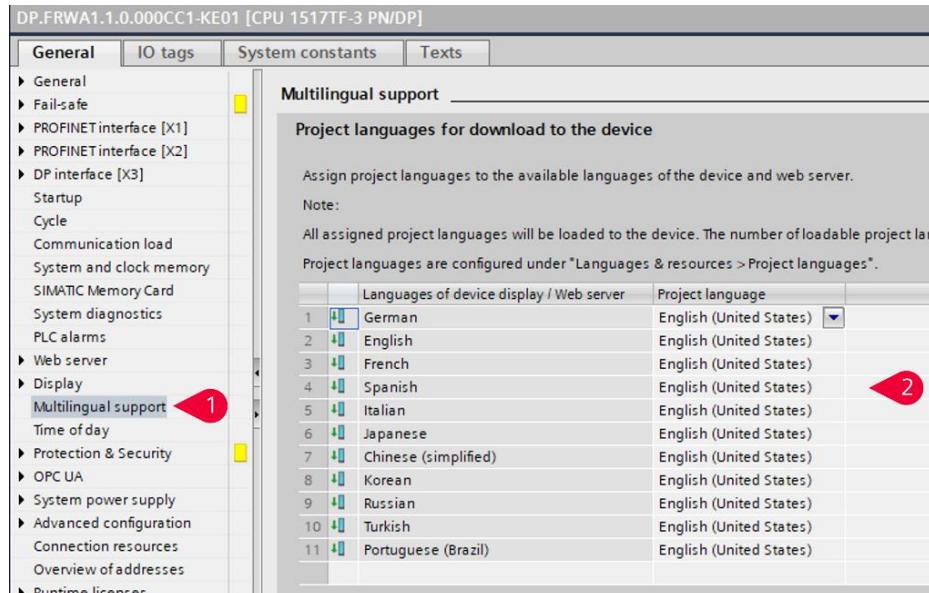


图 15-4 项目语言分配

所有分配的项目语言都将下载到 PLC 中。最多可以同时向 PLC 下载三种不同的项目语言。如果随后切换系统语言，则相应的项目文本将更改为指定的语言。

注

此外，还要注意 HMI 语言设置。

15.2.3. ProDiag 函数块

需要 ProDiag FB 来处理 ProDiag 监控。如果启用监控，则会自动创建第一个 ProDiag FB。

注

有关 ProDiag 的详细信息，请参见 TIA Portal 文档（第 12.26 章）。

<https://support.industry.siemens.com/cs/de/en/view/109826862>。

一个 ProDiag FB 可以处理多达 1000 个监控。要进行更多 ProDiag 监控，必须生成额外的 ProDiag FB。在这种情况下，必须将 ProDiag 监控分配给特定的 ProDiag FB 进行处理。

可以根据工厂的结构构建用户程序和监控消息的组成：关于监控的数量，符号名称和分配“FB 监控实例”。

为了实现这种结构，所有 ProDiag FB 都存储在相应单元和 EM 的文件夹中。生成块后，在编译软件时将生成实例数据块和组织块“ProDiag_OB”。所有 ProDiag FB 都在此 ProDiag OB 中调用。

同一模块中具有监控功能（如操作数监控）的功能块现在分配给 ProDiag 功能块。这样就可以对模块进行分层监控。

[诊断原理](#)一章详细介绍了 ProDiag 的诊断结构。

15.2.4. ProDiag 许可证 – PLC

ProDiag 需要许可证“SIMATIC ProDiag S7-1500”。使用的 ProDiag 许可证数量在 PLC 的属性中设置。其中还显示了组态的 PLC 总共使用了多少个 ProDiag 监控。

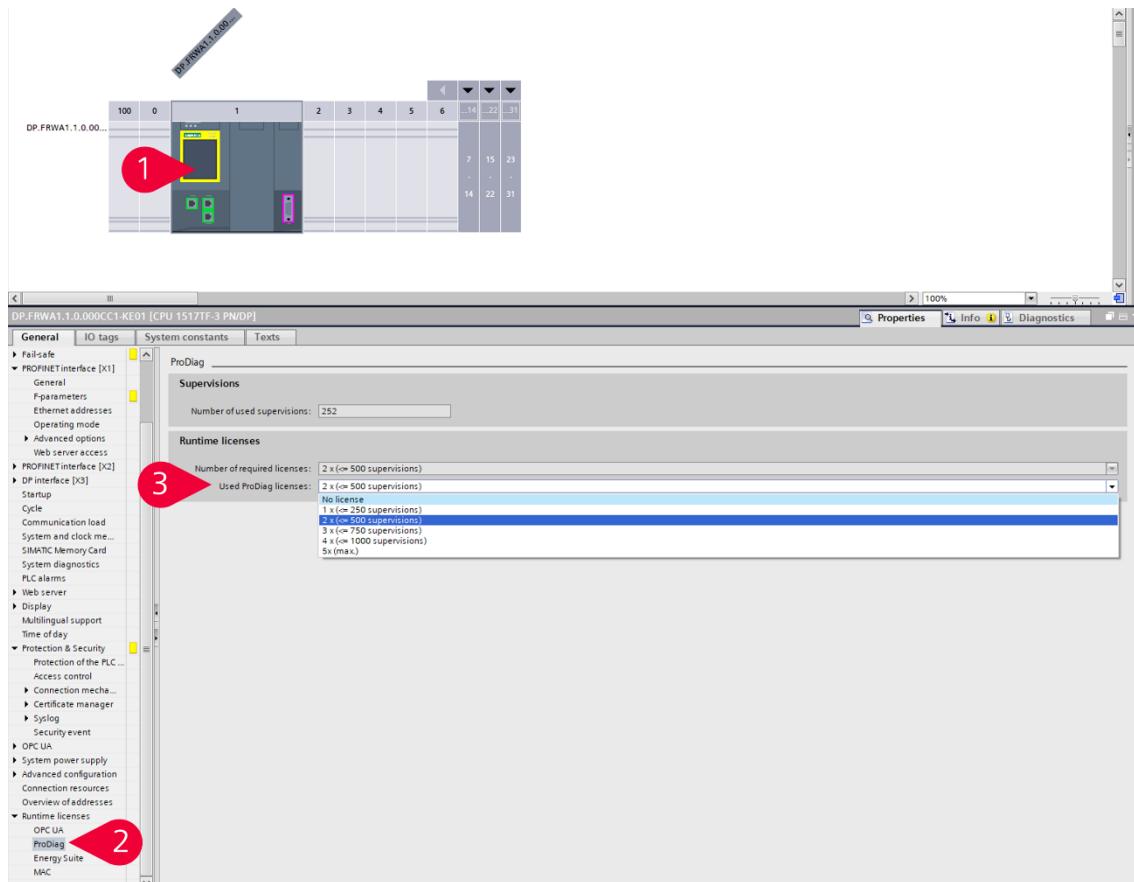


图 15-5 在 PLC 中组态许可证以实现 ProDiag 功能

15.2.5. 启用确认

出于诊断目的，即使触发条件不再处于活动状态，某些消息也需要保留。对于此类消息，必须组态专用确认。

在 ProDiag 块的属性中，打开选项卡“监控设置”，然后打开“类别”。输入分配给 HMI 确认按钮的复位/确认信号作为确认变量，用于类别“AVIAM、FV/FM”（报警、故障安全报警）。

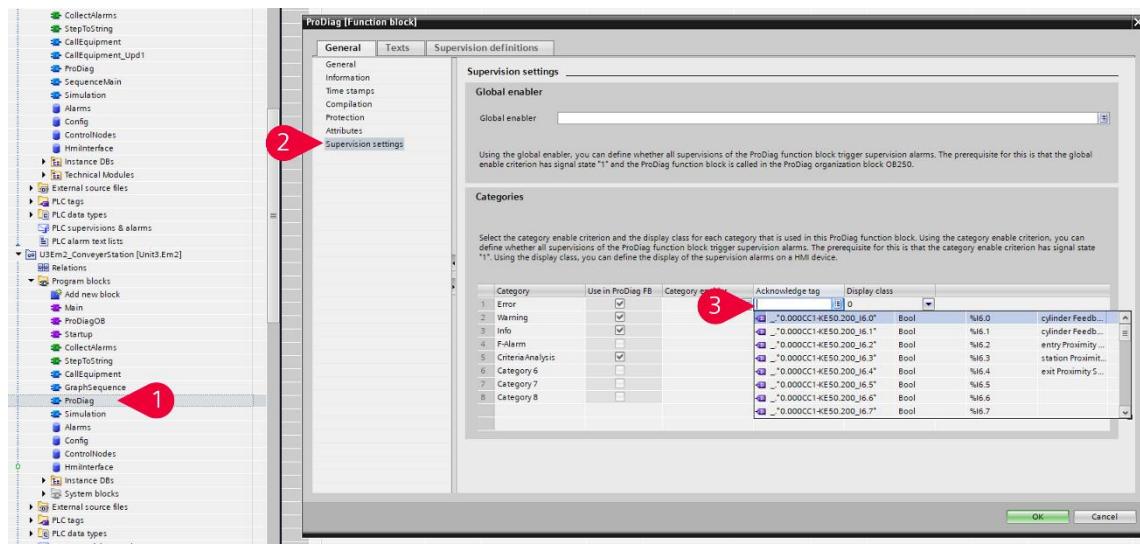


图 15-6 在函数块的 ProDiag 属性中激活确认变量

15.2.6. PLC 监控和报警

在“PLC 监控和报警”(1) 中，“全局监控”(2) 视图显示了现有全局监控的所有详细信息。在这里，可以找到系统集成商手动添加的所有监控。

“监控实例”(3) 视图显示所有通过添加并实例化功能块所产生的监控，这些监控被预先定义在功能块中。

Supervised tag	Trigger	ProDiag FB	ID
_Global.Data.general.enablePower	<input type="checkbox"/>	False	ProDiag [Global]
_Safety.DataFromSafety.eStopButton1.pressed	<input checked="" type="checkbox"/>	True	ProDiag [Safety]
_Safety.DataFromSafety.eStopButton2.pressed	<input checked="" type="checkbox"/>	True	ProDiag [Safety]
_Safety.DataFromSafety.eStopButton3.pressed	<input checked="" type="checkbox"/>	True	ProDiag [Safety]
_Safety.DataFromSafety.eStopButton4.pressed	<input checked="" type="checkbox"/>	True	ProDiag [Safety]
_Safety.DataFromSafety.safetyZone1.ackReq	<input checked="" type="checkbox"/>	True	ProDiag [Safety]
_Safety.DataFromSafety.safetyZone1.safetyReleased	<input type="checkbox"/>	False	ProDiag [Safety]
_Safety.DataFromSafety.lightCurtain1.interrupted	<input checked="" type="checkbox"/>	True	ProDiag [Safety]
_Unit1.Em0.Alarms.errorNoReaction[0]	<input checked="" type="checkbox"/>	True	ProDiag [Unit1.Em0]
_Unit1.Em0.Alarms.errorAbort[0]	<input checked="" type="checkbox"/>	True	ProDiag [Unit1.Em0]
_Unit1.Em0.Alarms.errorStop[0]	<input checked="" type="checkbox"/>	True	ProDiag [Unit1.Em0]
_Unit1.Em0.Alarms.warningNoReaction[0]	<input checked="" type="checkbox"/>	True	ProDiag [Unit1.Em0]
_Unit1.Em0.Alarms.warningHold[0]	<input checked="" type="checkbox"/>	True	ProDiag [Unit1.Em0]
_Unit1.Em0.Alarms.warningSuspend[0]	<input checked="" type="checkbox"/>	True	ProDiag [Unit1.Em0]
_Unit1.Em0.Alarms.information[0]	<input checked="" type="checkbox"/>	True	ProDiag [Unit1.Em0]
_Unit1.Em0.Alarms.criteriaAnalysis[0]	<input checked="" type="checkbox"/>	True	ProDiag [Unit1.Em0]
_Unit1.Em1.Alarms.errorNoReaction[0]	<input checked="" type="checkbox"/>	True	ProDiag [Unit1.Em1]
_Unit1.Em1.Alarms.errorAbort[0]	<input checked="" type="checkbox"/>	True	ProDiag [Unit1.Em1]
_Unit1.Em1.Alarms.errorStop[0]	<input checked="" type="checkbox"/>	True	ProDiag [Unit1.Em1]
_Unit1.Em1.Alarms.warningNoReaction[0]	<input checked="" type="checkbox"/>	True	ProDiag [Unit1.Em1]
_Unit1.Em1.Alarms.warningHold[0]	<input checked="" type="checkbox"/>	True	ProDiag [Unit1.Em1]
_Unit1.Em1.Alarms.warningSuspend[0]	<input checked="" type="checkbox"/>	True	ProDiag [Unit1.Em1]
_Unit1.Em1.Alarms.information[0]	<input checked="" type="checkbox"/>	True	ProDiag [Unit1.Em1]
_Unit1.Em1.Alarms.criteriaAnalysis[0]	<input checked="" type="checkbox"/>	True	ProDiag [Unit1.Em1]

图 15-7 PLC 监控和报警组态

15.2.7. 常见报警类别设置

对于 TIA Portal 中的每个项目，除了标准报警类别“确认”和“无确认”之外，还可以创建专门定义的报警类别。可以为新定义的报警类别设置名称、显示名称、优先级以及是否必须确认报警。

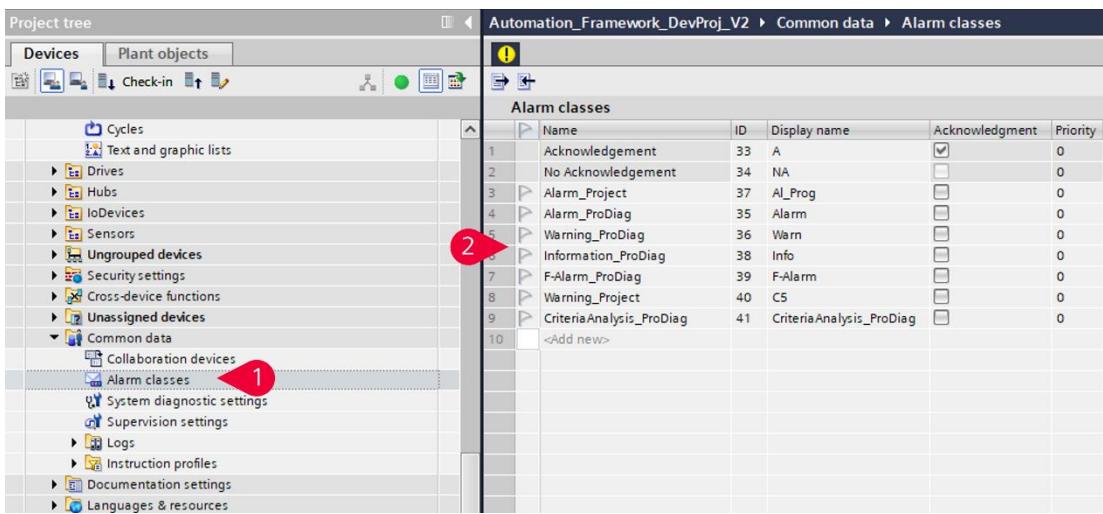


图 15-8 AF 报警类别组态

AF 提供以下用户自定义的报警类别:

- Alarm_Project
- Alarm_ProDiag
- Warning_ProDiag
- Information_ProDiag
- F-Alarm_ProDiag
- Warning_Project
- CriteriaAnalysis_ProDiag

然后, 可以使用报警类将分配给以下章节中描述的诊断设置。

15.2.8. 系统诊断设置

PLC 已经实现了某些诊断功能。可以将先前定义的报警类别分配给系统诊断的不同类别。为此, 首先在通用数据文件夹 (1) 中选择系统诊断设置区域, 然后调整分配 (2)。此外, 还可以停用项目的相应类别。

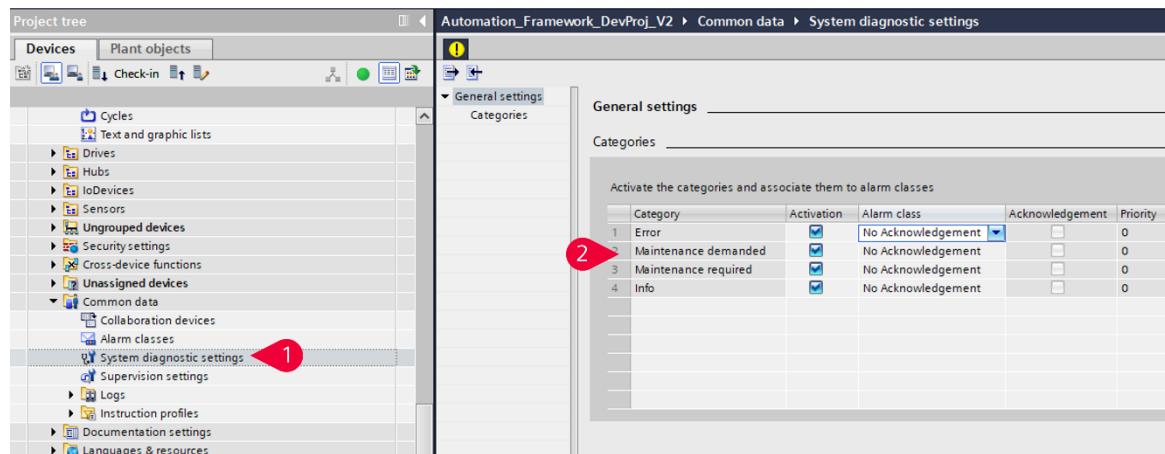


图 15-9 AF 中的系统诊断设置

15.2.9. 监控设置

诊断的最终项目级组态是所使用的监控的设置。在公共数据 (1) 中选择监控设置。常见的报警类别被分配给监控类别 (2)。有八个类别可用, 可以相应地激活和命名。此外, 还可以在两个子类别组中定义多达 16 个其他子类别 (3)。如何在 AF 中使用它们将在[诊断原理](#)一章中解释。

此处还定义了报警文本的结构 (4)。文本可以由不同的参数组成，并针对 GRAPH、基本监控、带错误消息的监控和带文本消息的监控单独定制。还可以以不同的方式显示全局创建和实例化的监控 (5)。

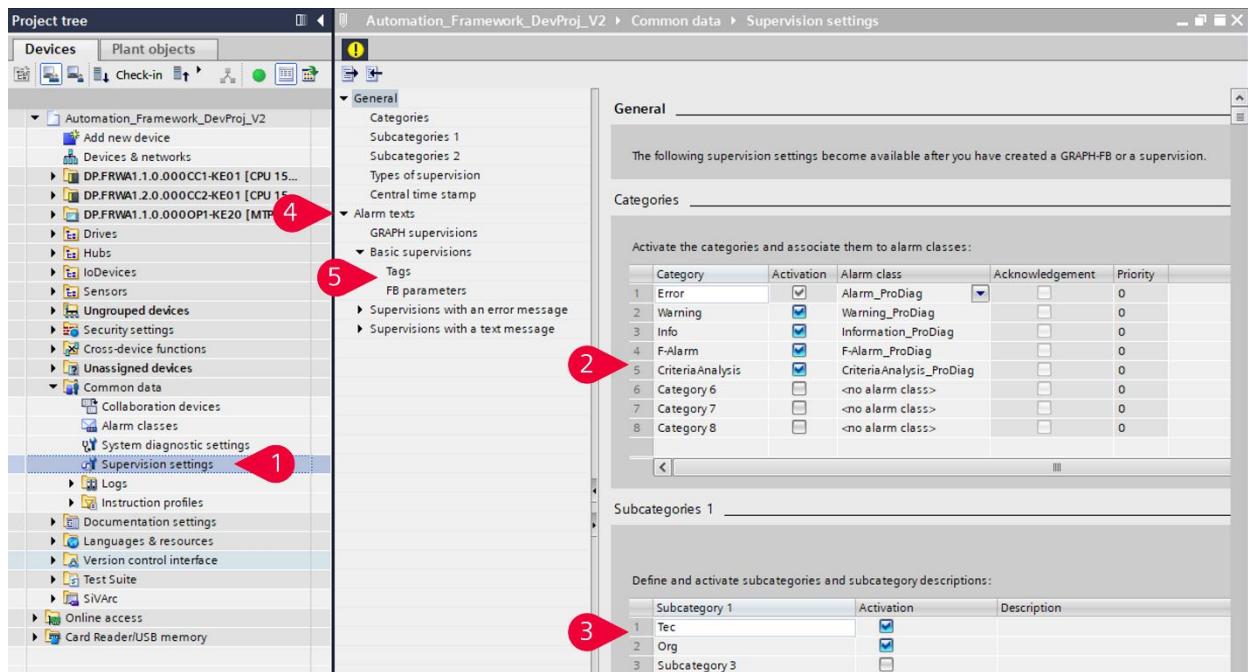


图 15-10 AF 内的监控设置

AF 的报警文本组态如下图所示（全局变量的监控 (1)、FB 参数的监控 (2)）：

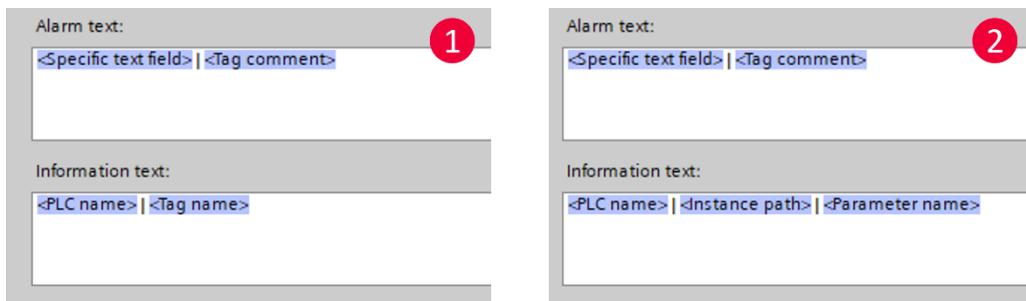


图 15-11 监控报警文本设置

15.2.10. HMI 报警类别设置

HMI 报警也可以自由组态，并且可以分配创建的报警类别。根据报警的状态，可以设置不同的颜色。通过组态的可视化设备(1)中的 HMI 报警，可以使用报警类别选项卡(2)访问组态概览(3)。



图 15-12 AF 的 HMI 报警设置

AF 的以下默认背景颜色设置为：

- 报警：红色
- 警告：橙
- 信息：蓝色
- F-报警：黄色

15.2.11. 自定义监控组态

监控是直接在 FB 的接口描述中创建的，或者是在全局 DB (1) 中创建的。在“属性”区域 (2) 中，可以进行监控组态 (3)。在创建 ProDiag 监控时，必须选择监控设置（类别、子类别 1、子类别 2）(4)。某些设置直接派生自监控类型和全局监控设置。以下章节详细介绍了如何实现 AF 内部监控的组态。

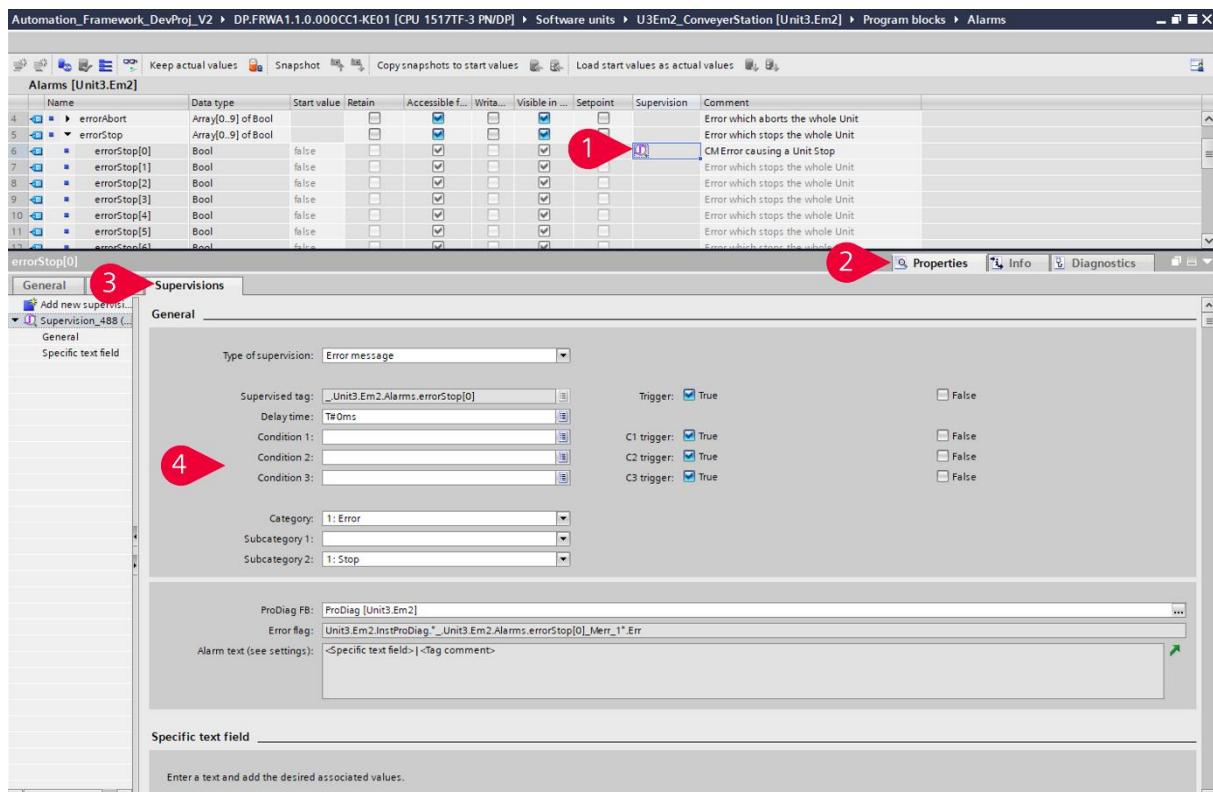


图 15-13 为 AF 自定义 ProDiag 设置

15.3. 诊断原理

AF 中的整体报警概念为在机器中创建报警提供了一种标准化和模块化的方法。其背后的原因是使操作员能够识别技术系统中任何故障的原因和位置。这些可以提高生产可用性并减少停机时间。

在 AF 中，这意味着需要一个定义明确的结构，概述如何将报警概念的关系集成到框架的功能架构中。这包括对 ISA-88 模型的不同层（从 CM 到单元级）的报警。

实现的报警概念由两个用于机器报警的组件组成。第一个组件涉及默认情况下已包含该概念的块中的预定义报警，例如 CM 层级的 LBC。第二个组件是 CollectAlarms 块等块，它允许轻松定义唯一报警。还可以监控与步序列相关的特定数据，并将这些监控包含在诊断概念中。

这些不同的监控用于将不同的报警映射到机器的过程，使它们能够直接影响机器的行为。此过程由

“LAF_ManageAlarms” 块处理。该块设置不同层的接口信号，以影响 LPML 单元模式状态管理器。下图显示了通过实施诊断概念，每个级别对更高实例的影响。所有以黄色或紫色显示的块都是库对象，不需要任何进一步的编程。

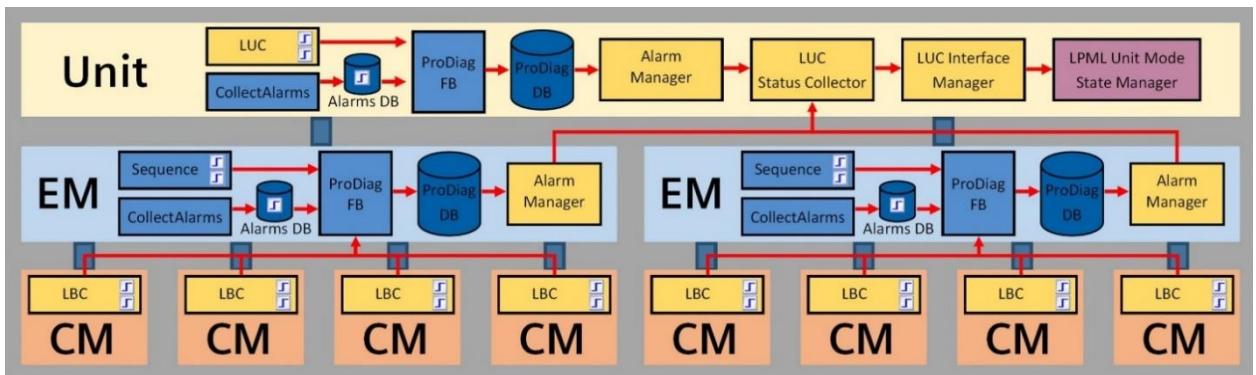


图 15-14 报警概念的架构

15.3.1. 报警组态

AF 为监控提供了不同的类别和子类别，以定义报警如何影响过程。为此，在项目中创建了五个用户特定的报警类别，并将其分配给过程诊断的主要类别 C1-C5。[监控设置](#)和[常见报警类别设置](#)章节中介绍了编辑这些设置的过程。此外，还有五个子类别用于表示报警对过程的影响。

报警类别	类别	子分类
Alarm_ProDiag	错误 (C1)	停止 (SC2.1)
Warning_ProDiag	警告 (C2)	继续 (SC2.2)
Information_ProDiag	信息 (C3)	中止 (SC2.3)
F-Alarm_ProDiag	F 报警 (C4)	保持 (SC2.4)
CriteriaAnalysis_ProDiag	条件分析 (C5)	挂起 (SC2.5)

表 15-1 诊断概念的监控设置

将类别和子类别分配给监控会影响报警的显示方式以及事件对设备过程的影响程度。

15.3.2. 报警实现

要实现不同级别的特定报警，可以使用“CollectAlarms”块。要添加报警，必须执行以下步骤：

- 定义应触发报警的条件，并根据报警的类别和子类别将其放置在“CollectAlarm”块的部分中。
- 插入分配并连接数组的一个空闲元素，该元素与“Alarms”DB 中的类别和子类别的组合相关。

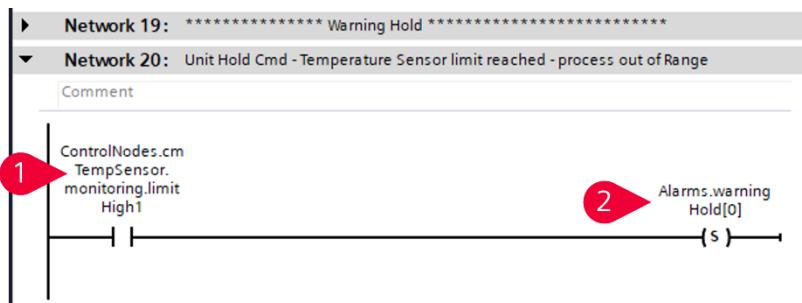


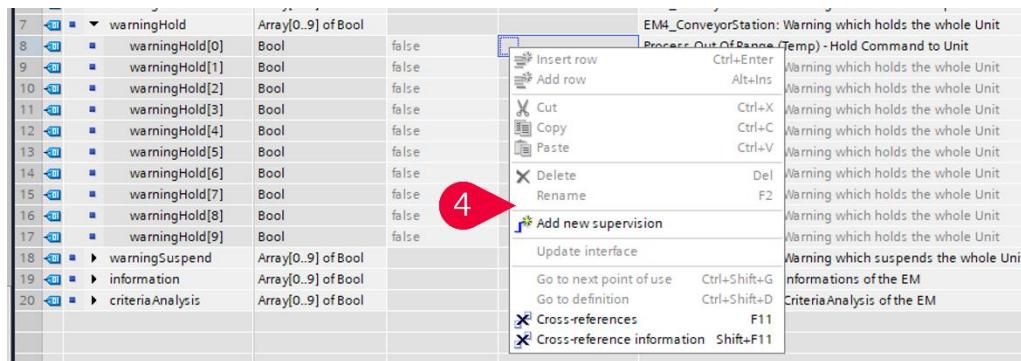
图 15-15 分配要监控的信号 (FC CollectAlarms)

- 打开“Alarms”DB，选择分配的元素并添加描述报警的备注。可以为已为项目定义的所有语言添加备注。



图 15-16 备注定义

- 向变量添加新的监控。如果同一数组中已存在监控，则可以使用复制和粘贴将其传输到所需的元素，并且可以跳过后续步骤。



7	warningHold	Array[0..9] of Bool		EM4_ConveyorStation: Warning which holds the whole Unit
8	warningHold[0]	Bool	false	Process Out Of Range Temp - Hold Command to Unit
9	warningHold[1]	Bool	false	Warning which holds the whole Unit
10	warningHold[2]	Bool	false	Warning which holds the whole Unit
11	warningHold[3]	Bool	false	Warning which holds the whole Unit
12	warningHold[4]	Bool	false	Warning which holds the whole Unit
13	warningHold[5]	Bool	false	Warning which holds the whole Unit
14	warningHold[6]	Bool	false	Warning which holds the whole Unit
15	warningHold[7]	Bool	false	Warning which holds the whole Unit
16	warningHold[8]	Bool	false	Warning which holds the whole Unit
17	warningHold[9]	Bool	false	Warning which holds the whole Unit
18	▶ warningSuspend	Array[0..9] of Bool		Warning which suspends the whole Unit
19	▶ information	Array[0..9] of Bool		Informations of the EM
20	▶ criteriaAnalysis	Array[0..9] of Bool		CriteriaAnalysis of the EM

图 15-17 在报警 DB 中创建监控

- 通过选择匹配的类别和子类别来组态监控。

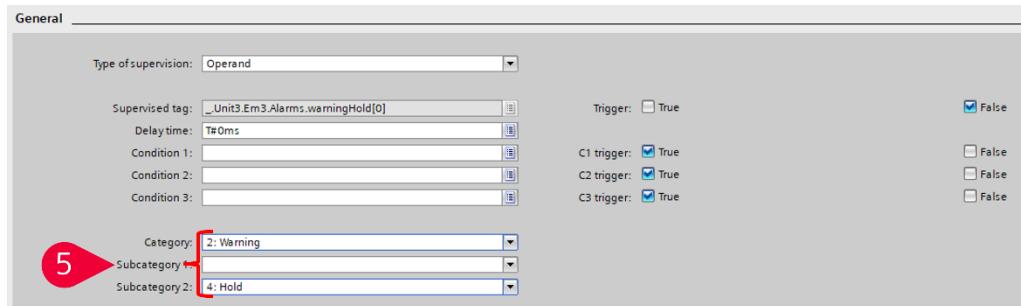


图 15-18 监控类别和子类别定义

- 在特定文本字段区域的 Tag 1 (SD_4) 输入处连接级别的参照代号。
- 在特定文本字段中定义 Tag 1 (SD_4) 的用法。

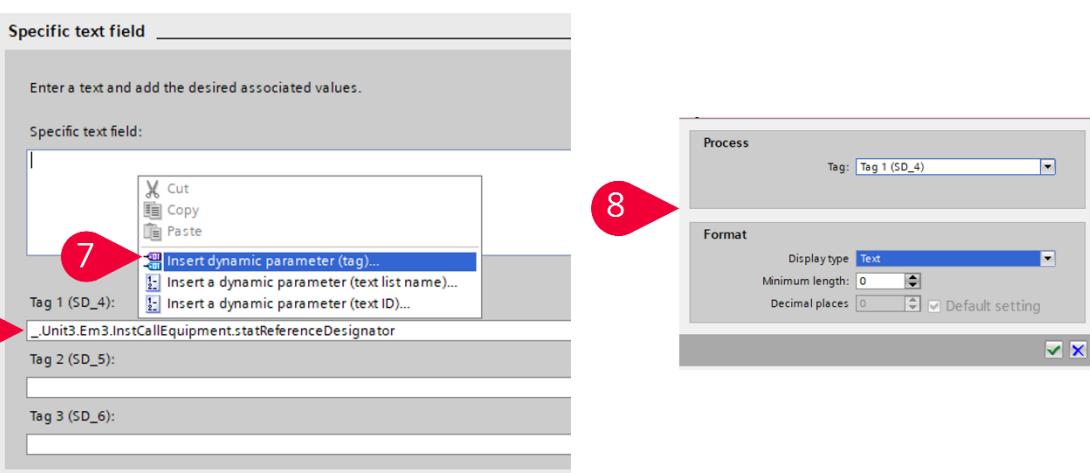


图 15-19 特定文本字段组态

也可以直接在要监视的信号上创建监控，并定义相应的类别和子类别。分析过程时必须考虑正确的报警定义。[图 15-20：在被监控信号本身创建监控](#) 显示了如何直接在被监控信号中实施报警，而无需将报警映射到“Alarms” DB。

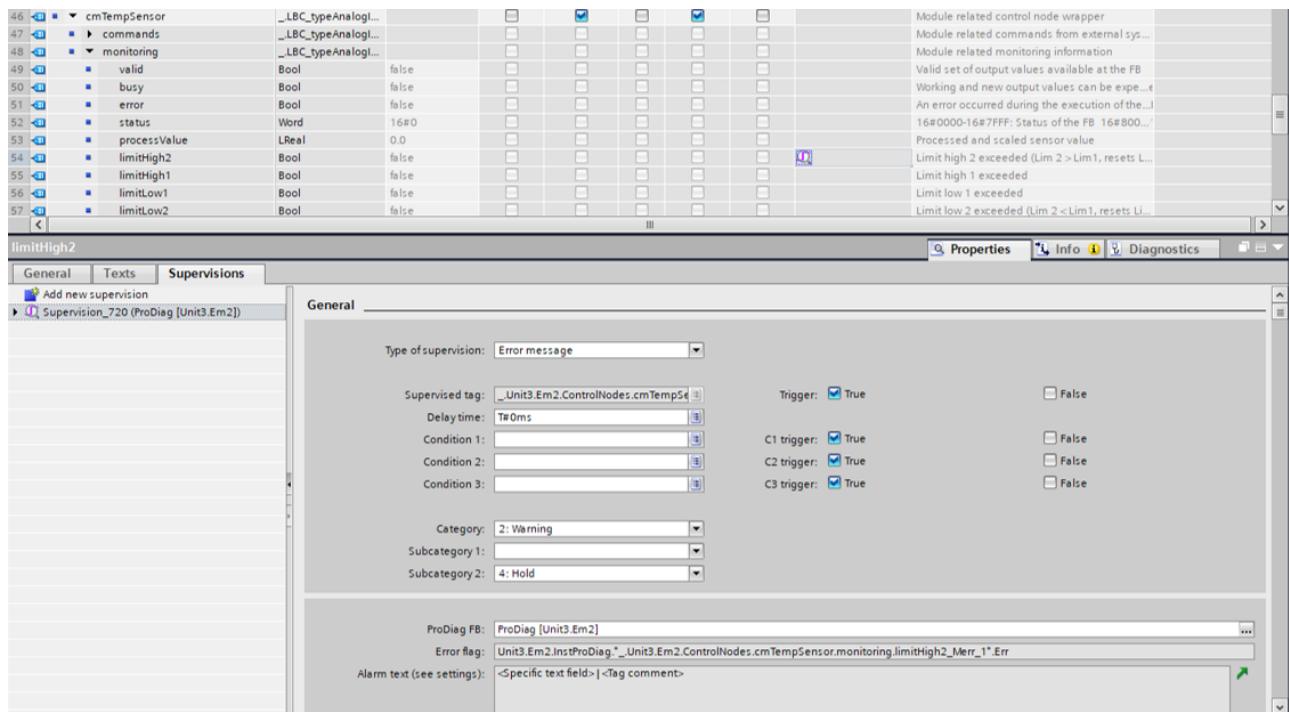


图 15-20：在被监控信号本身创建监控

如果已经存在对信号的监控，例如 LBC 块的情况，则只能在“CollectAlarm”块中添加对过程的反应。如果条件为真，将向单元发出指令，但不会从“Alarms”DB 中触发额外警报。

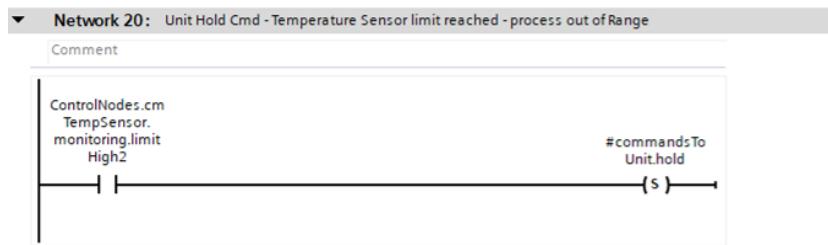


图 15-21 向已受监控的变量添加反应

15.3.3. LBC 库块的错误处理

错误检测

LBC 块具有内部错误检测和分析功能，以便为用户提供具有当前原因的状态。通过处理错误分析，用户更容易找到错误源和解决方案。

当发生错误时，应生成报警，并显示正确的报警信息。如果发生错误（例如，块操作中的错误），则必须输出错误，并设置状态（例如 16#8001）。常量被分配给状态代码，其中包含有关错误的特定信息（例如 16#8001：错误：函数块操作错误）。

这些常量包含在 PLC 报警文本列表 LBC_AxisControl_TecPlc。

Name	Data type	Default value	Retain	Accessible f...	Writ...	Visible in ...	Setpoint	Supervision	Comment
90	Constant								No call of FB
91	STATUS_NO_CALL	Word	16#7000						First cycle of FB after enabling
92	STATUS_FIRST_CALL	Word	16#7001						FB enabled
93	STATUS_SUBSEQUENT...	Word	16#7002						Invalid basic command selected (rising...
94	ERR_INVALID_BASIC...	Word	16#8001						Invalid extended command selected (ri...
95	ERR_INVALID_EXTEND...	Word	16#8002						Invalid superimposed command select...
96	ERR_INVALID_SUPERI...	Word	16#8003						Error occurred during camin command ...
97	ERR_INVALID_CAM...	Word	16#8200						Invalid jog mode selected
98	ERR_INVALID_JOG_M...	Word	16#8201						Command Error: invalid extended homi...
99	ERR_INVALID_HOMIN...	Word	16#8202						Error during extended homing comman...
100	ERR_TORQUE_LIMIT...	Word	16#8203						Invalid direction selected (jog forward a...
101	ERR_INVALID_JOG_DIR	Word	16#8204						Invalid direction is set for relative positi...
102	ERR_INVALID_POS_REL...	Word	16#8205						Invalid direction is set for superimpose...
103	ERR_INVALID_POS_SU...	Word	16#8206						Passive/Direct homing is not allowed d...
104	ERR_INVALID_HOMING...	Word	16#8207						Extended homing mode (homing on fix...)
105	ERR_MODULO_NOTE...	Word	16#8208						Configured torque limit value for extend...
106	ERR_INVALID_TORQUE...	Word	16#8209						Configured velocity in (technology obje...
107	ERR_INVALID_VELOCITY	Word	16#820A						No axis interconnected
108	ERR_INVALID_AXIS	Word	16#820B						Error occurred during MC_POWER comm...
109	ERR_MC_POWER	Word	16#8600						Error occurred during MC_RESET comm...
110	ERR_MC_RESET	Word	16#8601						Error occurred during MC_HOME comm...
111	ERR_MC_HOME	Word	16#8602						Error occurred during MC_TORQUELIMIT...
112	ERR_MC_TORQUELIMIT...	Word	16#8603						Error occurred during MC_HALT comma...
113	ERR_MC_HALT	Word	16#8604						Error occurred during MC_MOVEJOG co...
114	ERR_MC_MOVEJOG	Word	16#8605						Error occurred during MC_MOVEVELOCI...
115	ERR_MC_MOVEVELOCITY	Word	16#8606						Error occurred during MC_MOVERELATI...
116	ERR_MC_MOVERELATIVE	Word	16#8607						Error occurred during MC_ALARMTEXT...

图 15-22 LBC_AxisControl_TecPlc 块中错误处理的状态代码和常量

Name	Selection	Comment
LBC_AxisControl_StdPlc	Decimal	TextList for library block: LBC_AxisControl_StdPlc
LBC_AxisControl_TecPlc	Decimal	TextList for library block: LBC_AxisControl_TecPlc
LBC_Counter	Decimal	TextList for library block: LBC_Counter
LBC_DigitalSignal	Decimal	TextList for library block: LBC_DigitalSignal

Range from	Range to	Entry
28672	28672	No call of FB (code: 16#7000)
28673	28673	First cycle of FB after enabling (code: 16#7001)
28674	28674	FB enabled (code: 16#7002)
32769	32769	Invalid basic command selected (rising edge at 2 or more basic motion com...
32770	32770	Invalid extended command selected (rising edge at torque limiting and homi...
32771	32771	Invalid superimposed command selected (rising edge at 2 or more superimp...
32771	32780	Occurred during camin command (invalid cam connected at cam input) (code:...
32821	32821	Invalid jog mode selected (code: 16#8201)
32822	32822	Command Error: Invalid extended homing mode selected (code: 16#8202)
32823	32823	During extended homing command - torque limiting not allowed during homi...
32824	32824	Invalid direction selected (jog forward and jog backward) (code: 16#8204)
32825	32825	Invalid direction is set for relative positioning command (code: 16#8205)
32826	32826	Invalid direction is set for superimposed positioning command (code: 16#820...
32827	32827	Passive/Direct homing is not allowed during extended homing mode (homing ...)
32828	32828	Extended homing mode (homing on fixed stop process) is not allowed with m...
32829	32829	Configured torque limit value for extended homing mode (homing on fixed st...
32920	32920	Configured velocity (in technology object) for active homing = 0 (code: 16#82...
32921	32921	No axis interconnected (code: 16#8208)
34304	34304	Occurred during MC_POWER command (code: 16#8600)
34305	34305	Occurred during MC_RESET command (code: 16#8601)
34306	34306	Occurred during MC_HOME command (code: 16#8602)
34307	34307	Occurred during MC_TORQUELIMITING command (code: 16#8603)
34308	34308	Occurred during MC_HALT command (code: 16#8604)
34309	34309	Occurred during MC_MOVEJOG command (continuous jogging) (code: 16#860...
34310	34310	Occurred during MC_MOVEVELOCITY command (code: 16#8606)
34311	34311	Occurred during MC_MOVERELATIVE command (code: 16#8607)
34312	34312	Occurred during MC_MOVEABSOLUTE command (code: 16#8608)
34313	34313	Occurred during MC_MOVESUPERIMPOSED command (code: 16#8609)
34314	34314	Occurred during MC_GEARIN command (code: 16#860A)
34315	34315	Occurred during MC_PHASEGABSOLUTE command (code: 16#860B)
34316	34316	Occurred during MC_PHASESINGLERELATIVE command (code: 16#860C)
34317	34317	Occurred during MC_CANNIN command (code: 16#860D)
34318	34318	Occurred during MC_GEARINPOS command (code: 16#860E)
34319	34319	Occurred during MC_SYNCHRONIZEDMOTIONSIMULATION command (code: 16#8700)
34560	34560	Due to an undefined FB state (code: 16#8700)

图 15-23 PLC 报警文本列表中表示的 LBC_AxisControl_TecPlc 诊断信息

报警生成

LBC_AxisControl_TecPlc FB 输出区域中的错误位组态有 ProDiag 监控，状态参考相应的报警消息。

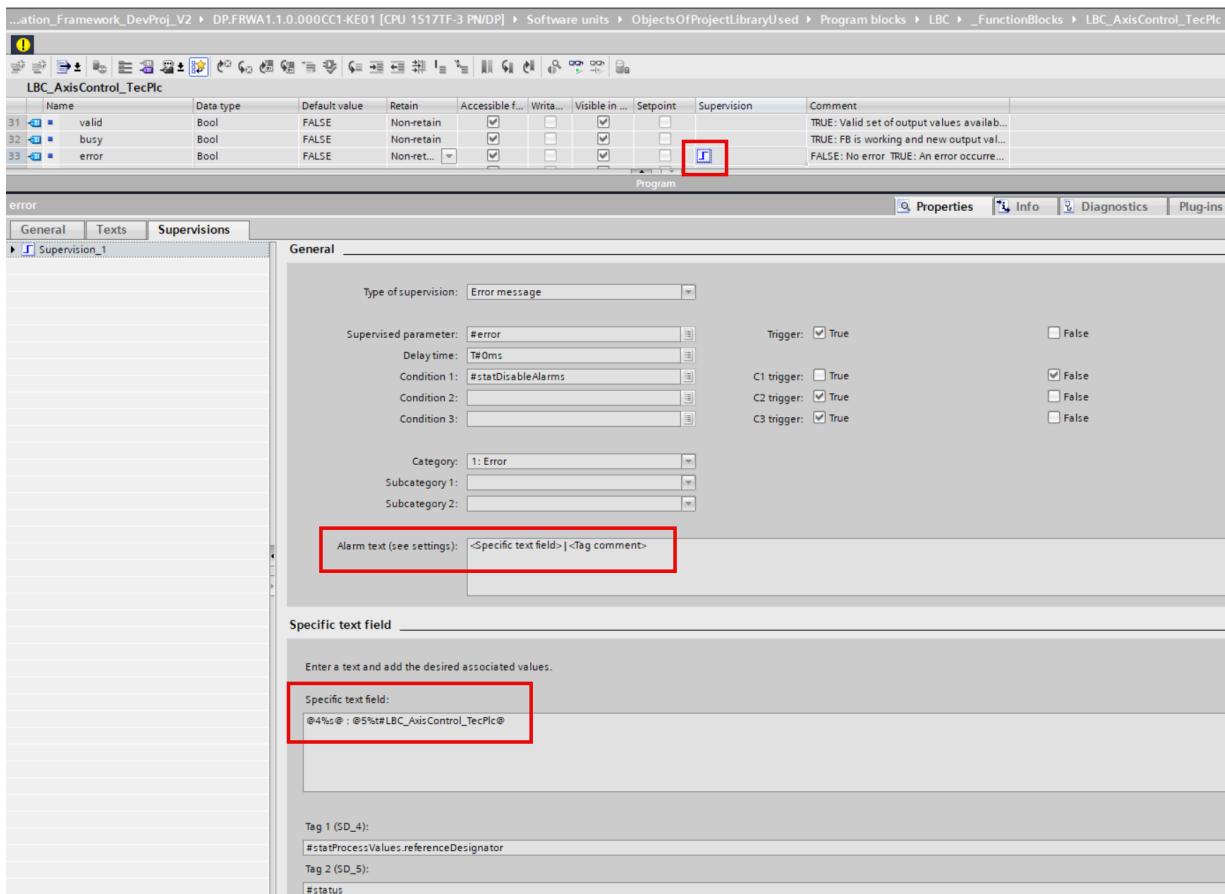


图 15-20 ProDiag 监控组态

在所选 ProDiag 监控的属性中，将显示有关报警文本的特定信息，并且可以进行组态。在此示例中，显示的报警消息包括特定文本字段和变量备注。

在特定文本字段中，将使用关联的值。在 AF 项目中，组态了动态参数（文本列表）。

<"Tag 1 (SD_4)":Text list:LBC_AxisControl_TecPlc >

报警消息是使用提供的文本列表结合 FB 的状态字生成的。

这些在 EM 中生成的报警消息通过 EM 状态 UDT 转发到单元，并通过 HMI 显示状态的变化。单元汇总了 ProDiag 生成的信号，并创建一个状态，状态中还包括 EM 是否耦合到单元。单元的安全状态也包含在此集合中。

注

有关 ProDiag 的详细信息，请参见以下条目
<https://support.industry.siemens.com/cs/ww/en/view/109740151>。

16. 仿真

16.1. 概览



S7-PLCSIM 产品:

S7-PLCSIM 产品及其技术特征和差异的比较如下:

<https://support.industry.siemens.com/cs/ww/en/view/109955578/173421632907>

注

使用 PLCSIM Advanced、SIMIT 和 HMI 仿真需要额外的软件包和许可证。更多详细信息请参阅“[硬件和软件要求](#)”一章。

使用 S7-PLCSIM 仿真 PLC

S7-PLCSIM 提供了对 PLC 程序已编写的逻辑的故障排除、验证和确认的工具。仿真表（Simulation table）允许监控和修改 PLC 输入和输出以及 DB 标签。仿真序列（Simulation sequence）可用于仿真与 PLC 程序交互的外部过程。此外，可以对功能块进行编程，以提供 PLC 内过程的预期行为。

这是一种手动测试 PLC 程序中逻辑的方法。仿真的质量取决于在 S7-PLCSIM 的仿真序列中建模过程的准确性，或在 PLC 内功能块编程的准确性。

利用 S7-PLCSIM Advanced 和 SIMIT 仿真系统行为

通过模拟传感器和执行器的外围行为和反应以及模拟过程的技术行为，对 PLC 程序进行全面验证和确认。在这种情况下，PLC 程序使用 S7-PLCSIM Advanced 进行仿真，外围设备和过程使用 SIMIT 进行仿真。

这种方法可以在开发阶段的早期发现潜在问题，并在不同的故障和错误情况下测试 PLC 程序的鲁棒性，而不会造成人员或材料损坏的风险。

不同仿真方式/工具的优势比较

优势	S7-PLCSIM	S7-PLCSIM Advanced with SIMIT	S7-PLCSIM Advanced with SIMIT
监控和调试 PLC 程序和变量	是	是	是
无需真实硬件的 PLC 程序的验证和确认	基础 PLC 逻辑，基本 I/O 模拟	高级 PLC 程序，基本 I/O 模拟	综合 PLC 程序、驱动器、传感器、执行器、安全等的外围行为
测试级别选项	单元测试 仅限于手动测试	集成测试 使用高级测试套件进行可选的自动单元测试	系统测试 SIMIT Rapid Tester 的可选自动模拟测试
验证与外部系统的通信，例如 MES、M2M 通信等	有限 最多两个 PLC 和 HMI 之间只有软总线	广泛 各种协议，安全通信	广泛 各种协议，安全通信

优势	S7-PLCSIM	S7-PLCSIM Advanced	S7-PLCSIM Advanced with SIMIT
仿真范围	基础 PLC 逻辑、数字和模拟 I/O	高级 PLC 程序、数字和模拟 I/O 通信	综合 复杂过程、机械相互作用、系统行为
故障和错误仿真	有限 PLC 程序中故障和错误程序的仿真	有限 PLC 程序中故障和错误程序的仿真	广泛 模拟故障和错误场景以测试系统的鲁棒性
过程仿真	否 应用解决方案	否 应用解决方案	是 过程行为的模拟，包括流体动力学、温度控制等
操作员/维护培训	有限 HMI 交互	有限 HMI 交互 Web 服务器	广泛 HMI 交互、Web 服务器、物理按钮等
与其他仿真工具集成	否 不适用	例如 SIMIT、Simcenter Amesim、Tecnomatix Process Simulate	是 例如 NX MCD、SINAMICS DriveSim Basic、Tecnomatix Process Simulate、Tecnomatix Plant Simulation
质量保证	低 PLC 逻辑和块中的错误检测	中 检测 PLC 程序和块中的错误，检测不正确的通信设置	高 在开发阶段早期检测潜在问题，降低实际调试期间的风险
设置复杂性	低 易于设置，无需额外培训	低 易于设置，无需额外培训	高 建立行为模型/数字孪生更复杂，需要额外的培训和专业知识来充分利用其能力
易用性	高	高	适中
成本	低 无需额外许可	中 需要额外的许可证	高 需要额外的许可证

表 16-1 不同仿真工具的优势比较

16.2. PLC 和 HMI 集成仿真信号

PLC 集成

所有与仿真的信号均由中央功能（Central Functions）软件单元中的 FC “CallSimulation” 集中管理，并通过全局数据分发到各单元。每个软件单元使用的仿真变量（tags）都组织在单独的变量表中，并在单独的功能块中编程，从而更容易从 PLC 程序中删除仿真。[图 16-1](#) 显示了中央功能、安全单元和设备模块“料斗”示例的程序架构。

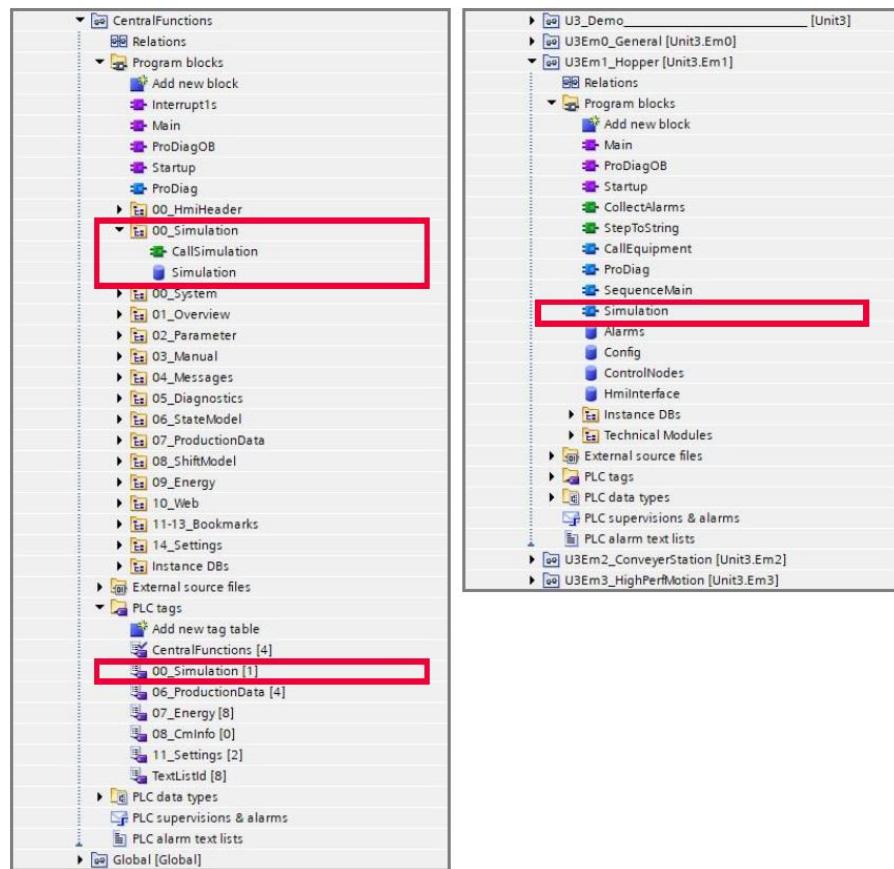


图 16-1 PLC 与仿真有关的 PLC 块和标签表

HMI 集成

全局数据的所有仿真变量也集中组织在一个 HMI 变量表中，这使得在 HMI 中删除仿真变得更加容易。默认情况下，HMI 变量链接到 HMI 对象，并在 AF 的不同 HMI 屏幕中实现。带有功能块的 PLC 仿真可以通过仿真按钮（1）激活或停用



图 16-2 PLC 与仿真相关的 HMI 对象和标签表

16.3. 使用 PLCSIM 进行 PLC 仿真

在 AF 示例项目中，PLC 程序中的仿真功能块默认被激活。当 PLC 仿真处于活动状态时，将类型轴上的工艺对象设置为仿真模式。按照以下步骤，我们可以将 PLC 仿真与仿真表和内部仿真功能块一起使用。

1、启动 S7-PLCSIM。

2、打开提供的预配置工作区（1）和（2）。选择上级文件夹“AF_PLCSim_Workspace_V1_2”（3）并打开它（4）。

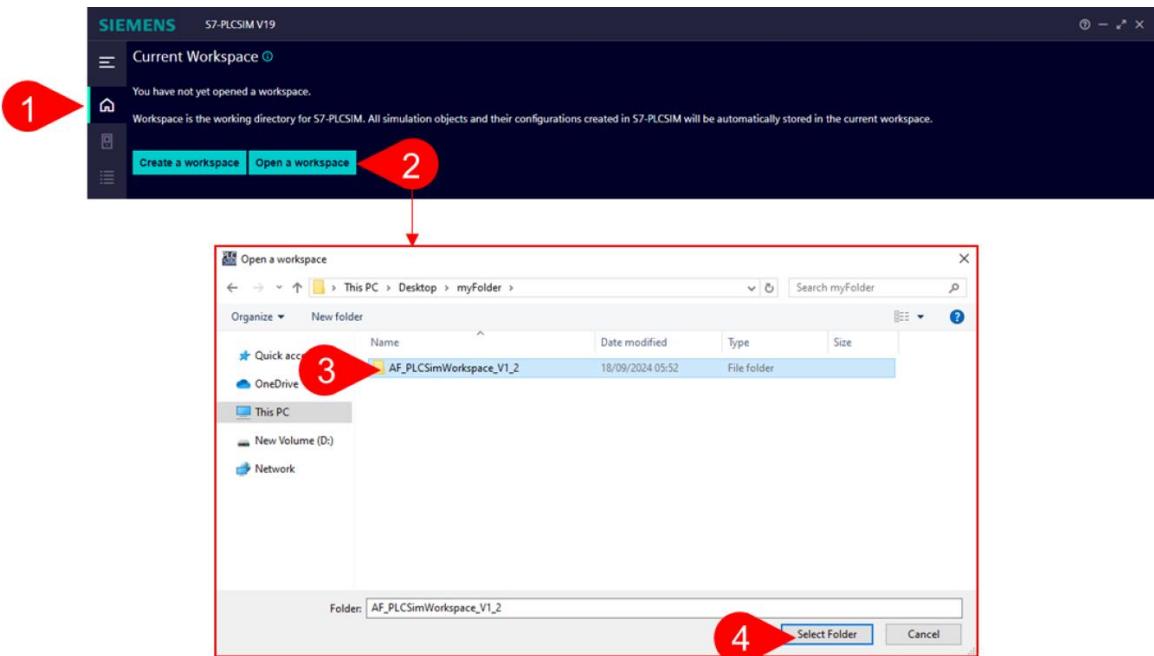


图 16-3 如何打开已保存的 PLCSIM 工作区

3、按照图 16-4 中的步骤启动模拟 PLC 实例。

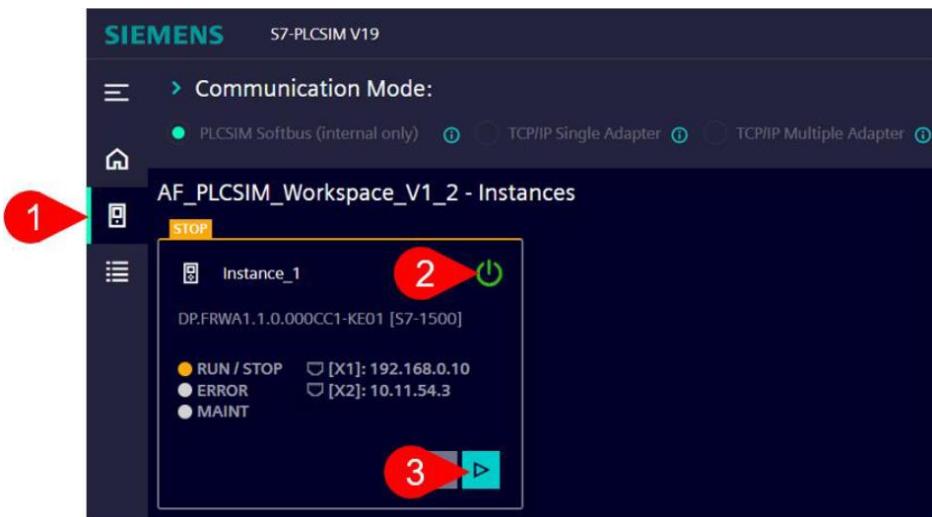


图 16-4 如何在 S7-PLCSIM 中启动模拟 PLC 实例

4、转到 TIA 门户并下载 PLC（1）。当仿真 PLC 实例当前正在运行时，PLCSIM 被预选为 PG/PC 接口（2）。在项目中进行更改时，此步骤是必要的，否则预配置的实例已经包含该项目，可以跳过此步骤。

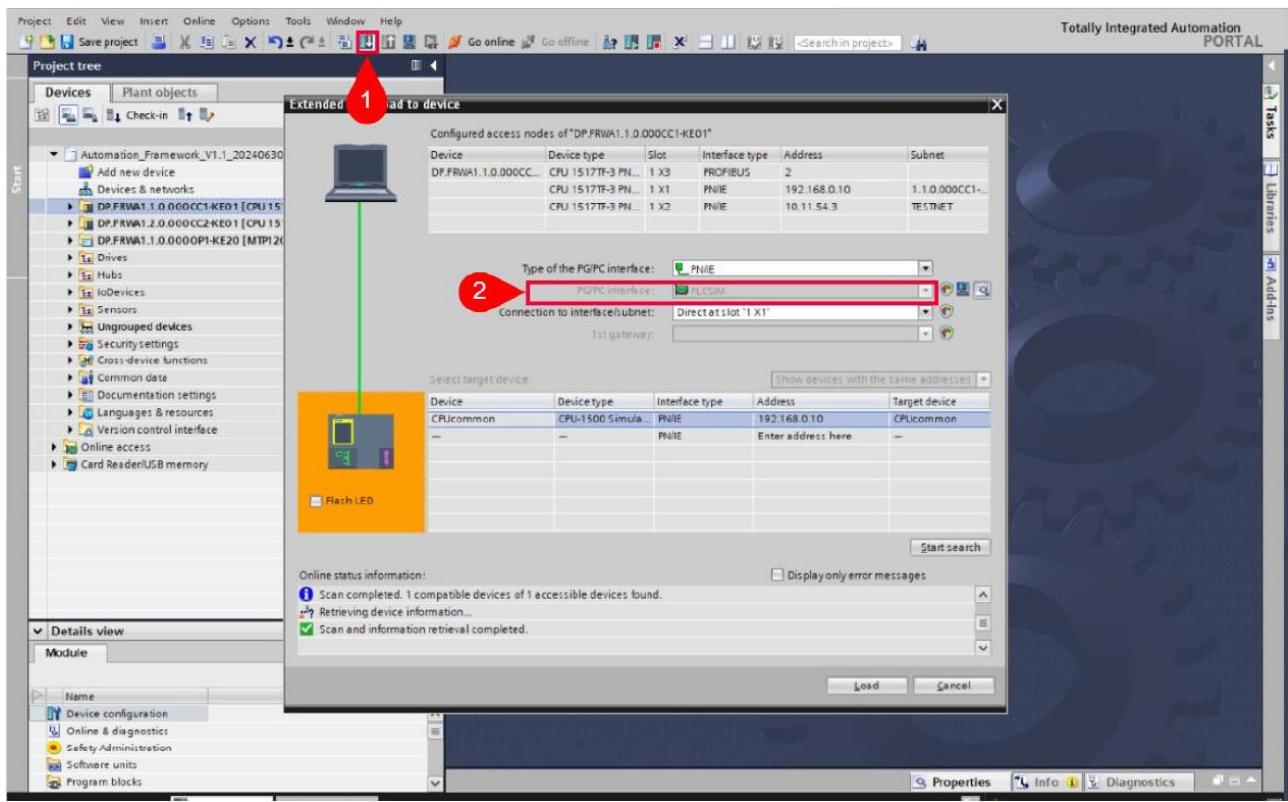


图 16-5 如何将 PLC 下载到模拟 PLC 实例中

5、可选：启动 HMI 仿真（1）。WinCC Unified runtime 将在浏览器中自动打开。HMI 仿真可能需要额外的软件和许可证，具体取决于 engineered power tags。

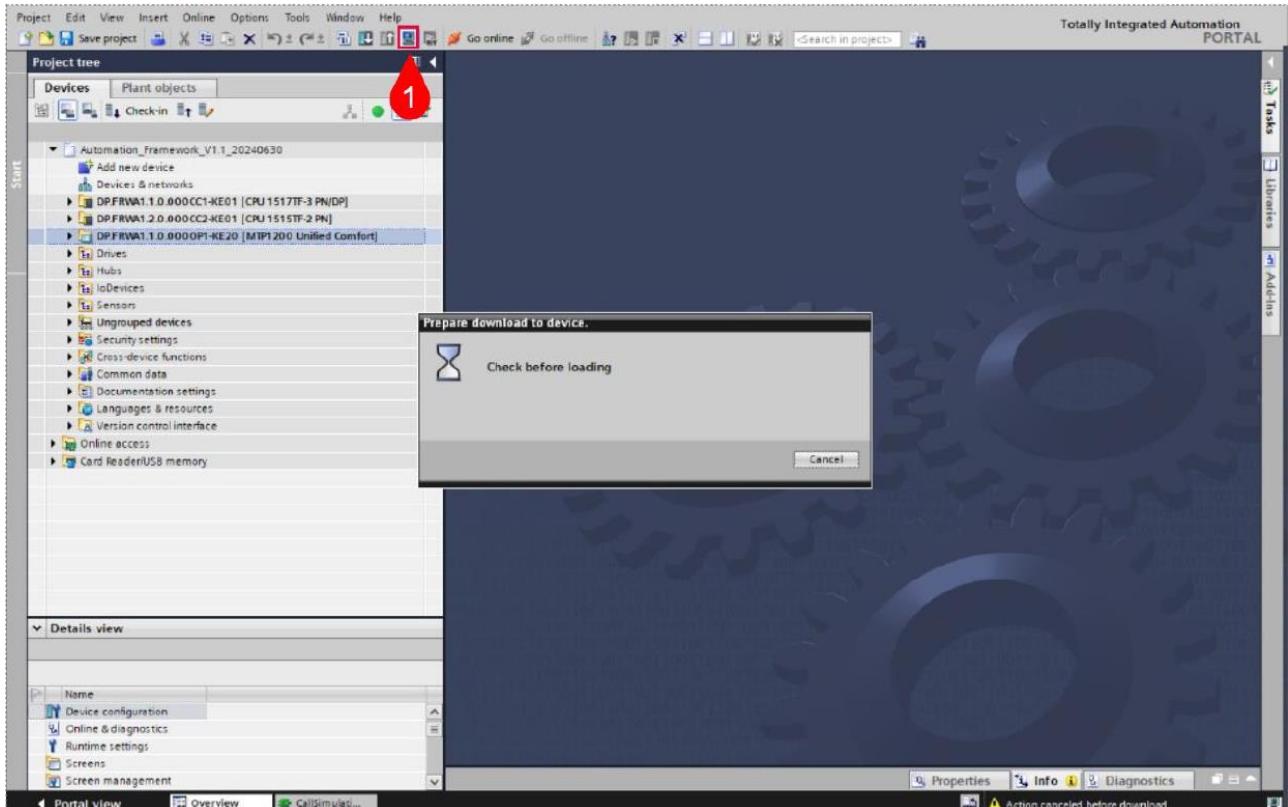


图 16-6 如何启动 HMI 应用程序的模拟

6、返回 S7-PLCSIM 并打开仿真选项卡（1）。仿真故障安全输入所需的变量（tags）已预先配置。启动仿真表以启用联机视图（2）。

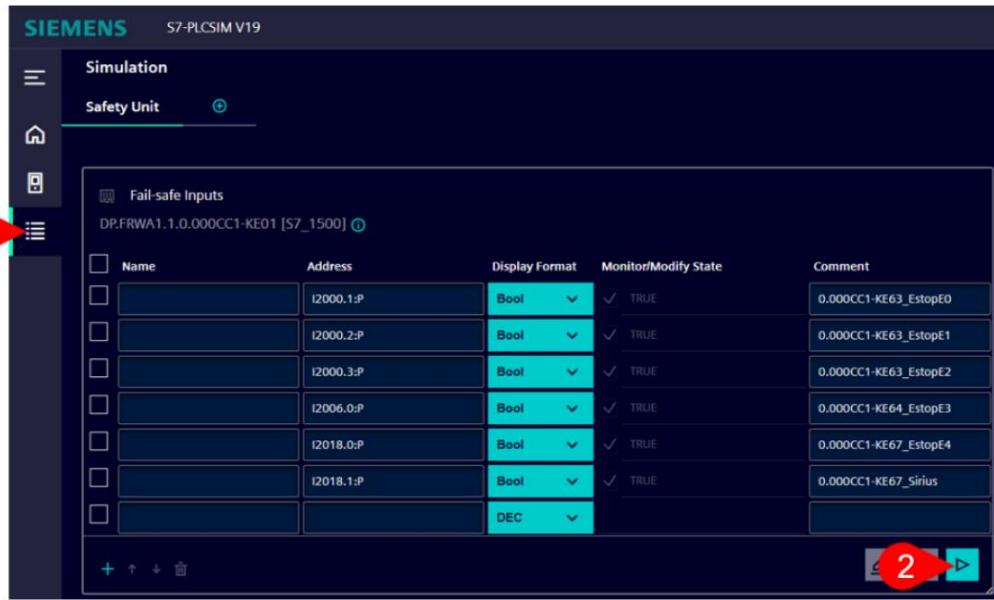


图 16-7 PLC 如何在 S7-PLCSIM 中启动仿真表

7、例如，通过修改仿真表（1）中的故障安全输入来测试和验证安全程序的逻辑。紧急按钮的故障安全输入状态设置为“False”（2 和 3），通过将 STO 值更改为“False（4）”，触发所有驱动器的 STO 功能。

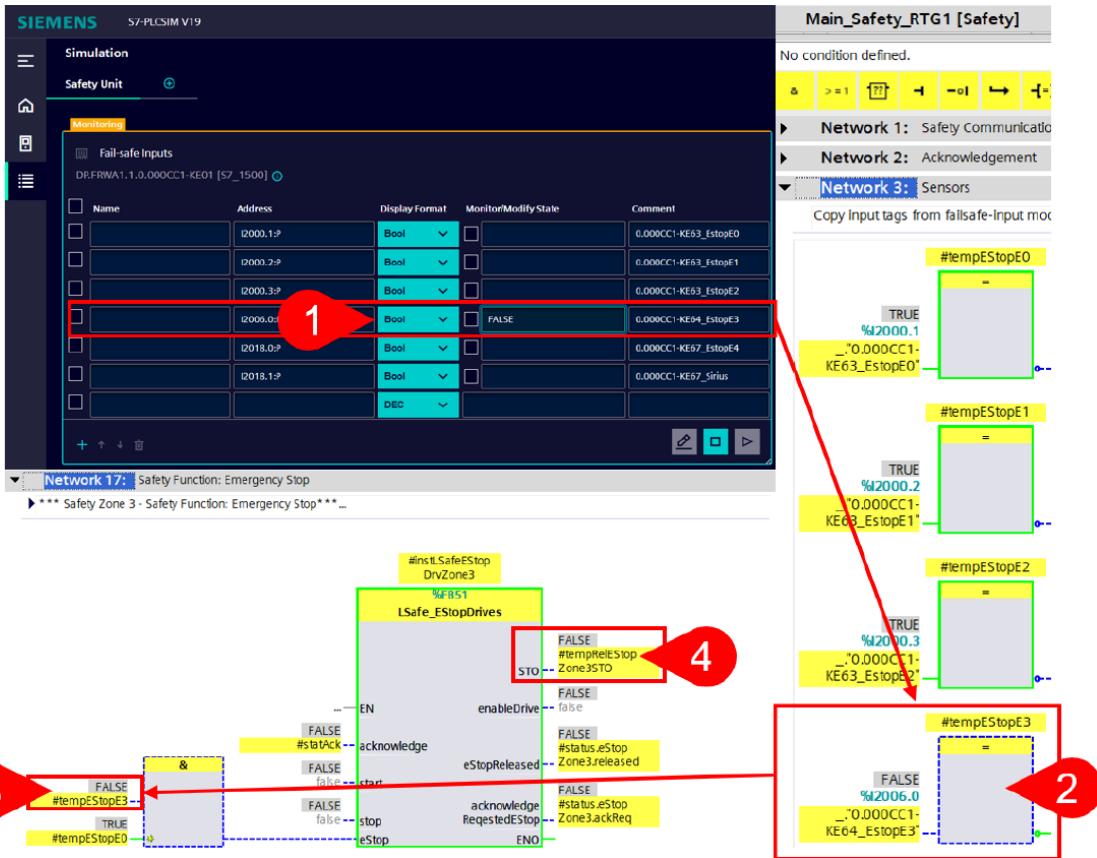


图 16-8 AF 仿真故障安全输入状态变化后安全程序中的反应

结果

安全程序的正确逻辑得到验证。然而，驱动器的正确响应并非如此。还必须手动模拟，以验证 PLC 程序中的响应。

16.4. 使用 SIMIT 仿真系统行为

注

所提供的示例 SIMIT 项目基于[演示单元](#)和附属演示设备模块[EM 演示实施-料斗](#)、[EM 演示实施 - 传送带](#)、[EM 演示实现 - 高性能运动](#)。

注

所提供的 SIMIT 项目在 S7-PLCSIM Advanced 的 PLCSIM 软总线中启动仿真 PLC 实例。在此模式下，OPC UA 服务器和 Web 服务器不可访问。为了测试通信，直接在 S7-PLCSIM Advanced 中以 TCP/IP 单适配器模式启动模拟 PLC 实例。

使用 SIMIT 时，建议在 SIMIT 项目中实现一个心跳信号（a life bit），该心跳信号连接到 TIA 项目中的数字输入。SIMIT 运行时，心跳信号始终设置为“TRUE”。此连接允许在 PLC 程序中触发响应（例如在 HMI 中显示信息消息，禁用 PLC 程序中的仿真 FB 等）。

在自动化框架的情况下使用 SIMIT 进行仿真时，[使用 PLCSIM 进行 PLC 仿真](#)被停用，HMI 对象（见图 16-2）被隐藏。此功能在软件单元“CentralFunctions”中的 FC “CallSimulation”中实现。按照以下步骤将 PLC 仿真与 SIMIT 一起使用：

- 1、启动 SIMIT SP 并打开提供的预配置项目。
- 2、通过播放按钮（1）启动模拟。S7-PLCSIM Advanced 和仿真 PLC 实例在后台自动启动。



图 16-9 如何在 SIMIT 中启动仿真

- 3、转到 TIA 门户并下载 PLC。当仿真 PLC 实例当前正在运行时，PLCSIM 被预选为 PG/PC 接口。
- 4、启动 HMI 的仿真。WinCC Unified runtime 将在浏览器中自动打开。HMI 仿真可能需要额外的软件和许可证，具体取决于 engineered power tags。
- 5、[图 16-10](#) 显示了 SIMIT 中[EM 演示实施-传送带](#)的运行仿真（2）。运行中的仿真以橙色突出显示。该仿真允许验证控制模块的 I/O（3）和功能块（例如 LBC_TwoWayActuator、LBC_AnalogInput（4）），验证 OMAC PackML 状态机的反馈信号或测试序列。

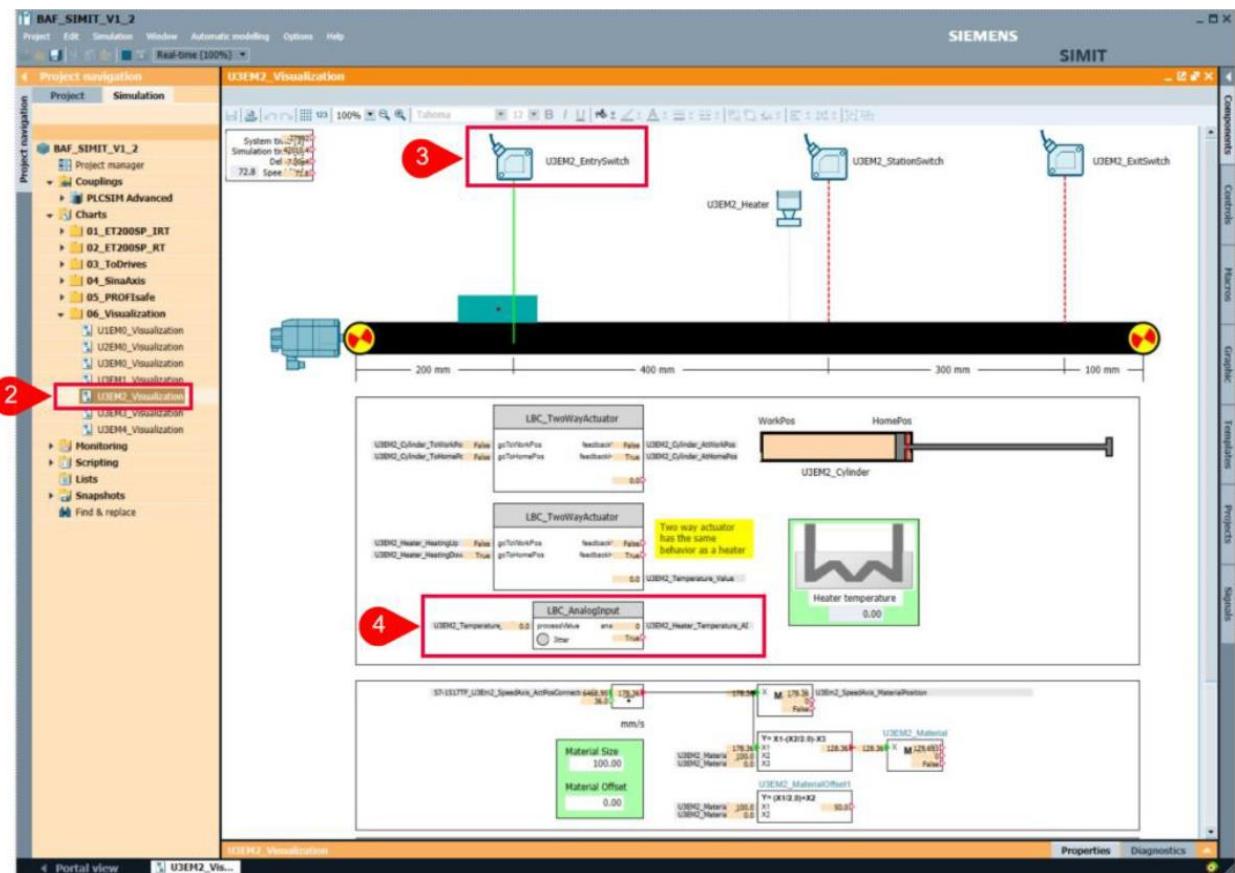


图 16-10 在 SIMIT 中运行仿真

6、[图 16-11](#) 显示了安全程序作为典型仿真案例的验证。按下紧急停止按钮（5）。正确的响应通过 PROFIsafe 报文 30 的行为模型进行验证。在这个例子中，按下紧急停止按钮后，STO 的值变为“FALSE”，驱动器的断电状态响应为正确的值“TRUE”（6）。

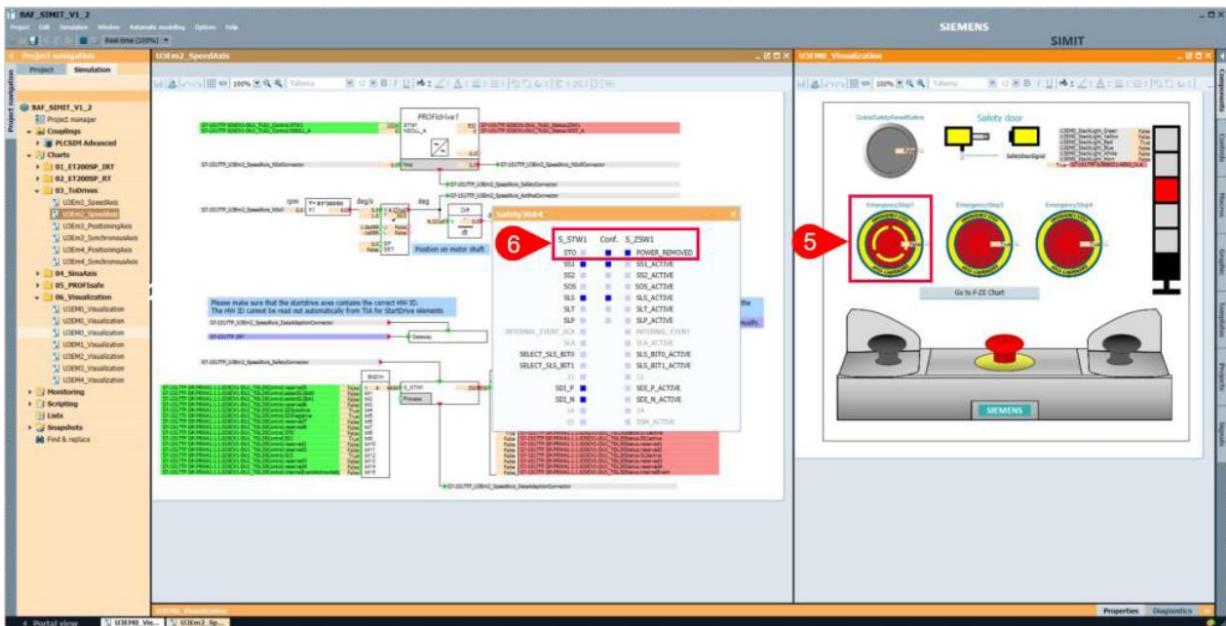


图 16-11 仿真案例——紧急停止触发的驱动器 STO 响应的安全程序验证

结果

验证了安全程序的正确逻辑和基于驱动响应的 PLC 程序中的正确响应。

17. 用户管理

17.1. 概述

为了集中管理工厂网络中的用户和用户组，并使用域用户，TIA Portal 提供了一个随时可以运行的用户管理组件 (UMC)。TIA Portal 已经提供了一系列预组态的角色。客户可以添加其他角色并根据他的要求组态它们。

UMC 服务器中的用户/组组态可以导入到 TIA Portal 项目中，因此此组态是独立于设备数量的单个操作。

要管理项目中的用户，可以执行以下操作：

- 打开编辑器“Security settings > Users and roles” (1)。
- 打开角色窗口 (2)
- 选择一个用户 (3)，该用户的设置将显示在底部屏幕中
- 打开运行系统权限窗口 (4)，可以看到角色的运行系统权限 (5)。
- 要为用户分配角色，请按以下步骤操作：
- 打开“Users”选项卡 (6)。
- 选择要为其分配角色的用户 (7)。注意不能使用多重选择。
- 在“Assigned roles”部分 (8) 启用所需的角色 (9)。

所描述的过程如下图所示：

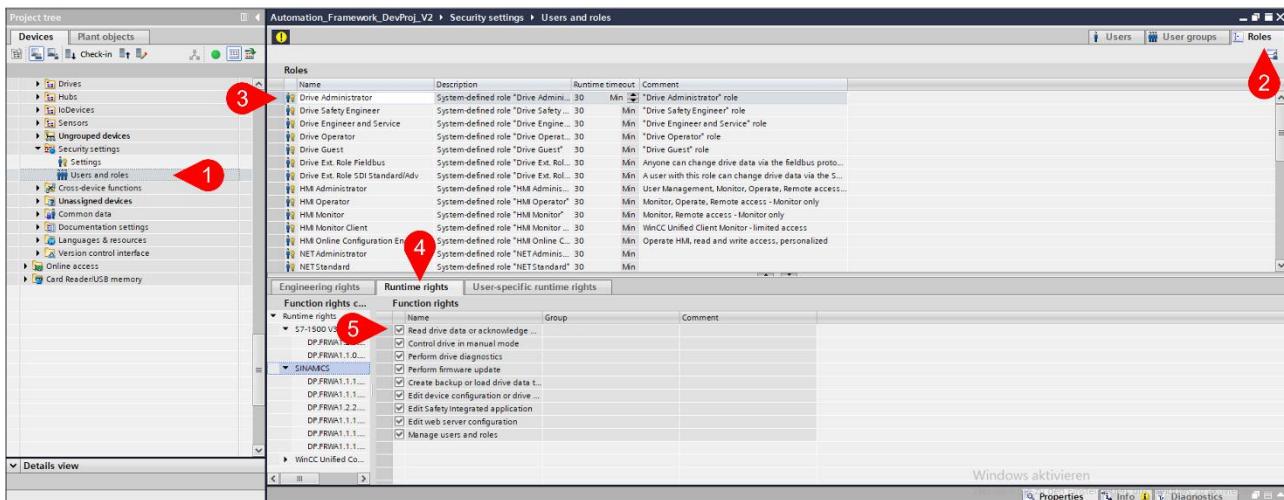


图 17-1 组态角色和权限

图 17-2 组态用户和分配的角色

使用 WinCC Unified Runtime 的“用户管理组件 (UMC)”进行中央用户管理

所有用户都可以使用他们的个人凭据登录，并可以按照他们的专用角色操作 HMI。

用户管理和访问控制 (UMAC)，用于保护项目、PLC、驱动器和 Scalance

TIA Portal 为 TIA 项目和设备提供全面的 UMAC。可以管理对工程组态的访问权限。用户只能基于其角色运行功能。

要激活 UMAC，必须启用项目保护。



项目保护不能撤销。

如果丢失受保护项目的登录信息，项目将丢失。

只有在妥善保管登录名和密码信息的情况下，才建议使用项目保护。

注

如果需要确保功能或组态不再被任何人更改，建议在调试阶段尽早激活 UMAC 或项目保护，最迟在移交给最终用户后。



请在 TIA Portal 的在线帮助中的“UMAC”下或 SIMATIC Step7 V19（第 9 章）手册中找到更多信息：

<https://support.industry.siemens.com/cs/de/en/view/109826862>。

17.2. 演示实现

AF 项目包括一个示例，展示了如何根据“Local Users”组态各种用户管理/访问控制。在“User-specific runtime rights”(4)中有三个不同的“function rights”，这些权限是为了给HMI用户设置不同的访问级别而创建的。下图中的管理员用户(3)，其“User-specific runtime rights”可在“Users and roles”菜单(1)中的“Roles”选项卡(2)中找到。

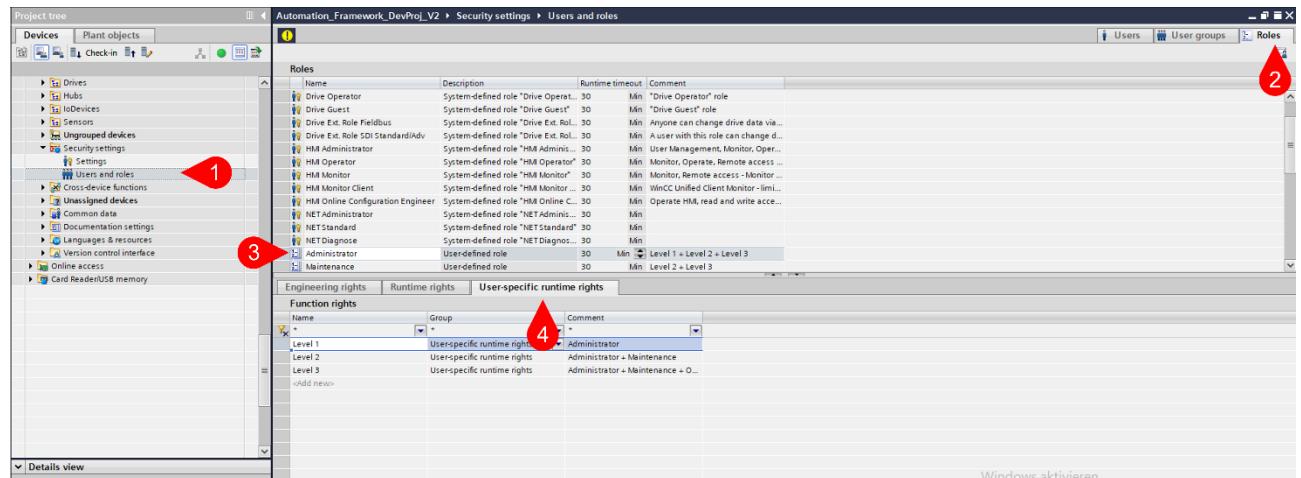


图 17-3 用户和角色页面

要为角色分配功能权限或删除已分配的功能权限，请按照以下步骤操作：

- 打开“Roles”选项卡。
- 选择角色(1)。请注意，分配时不能使用多重选择。
- 在下部区域，打开要分配或删除功能权限的类别选项卡。
- 激活要分配给角色的功能权限(2)。
- 停用不再分配给角色的功能权限(2)。

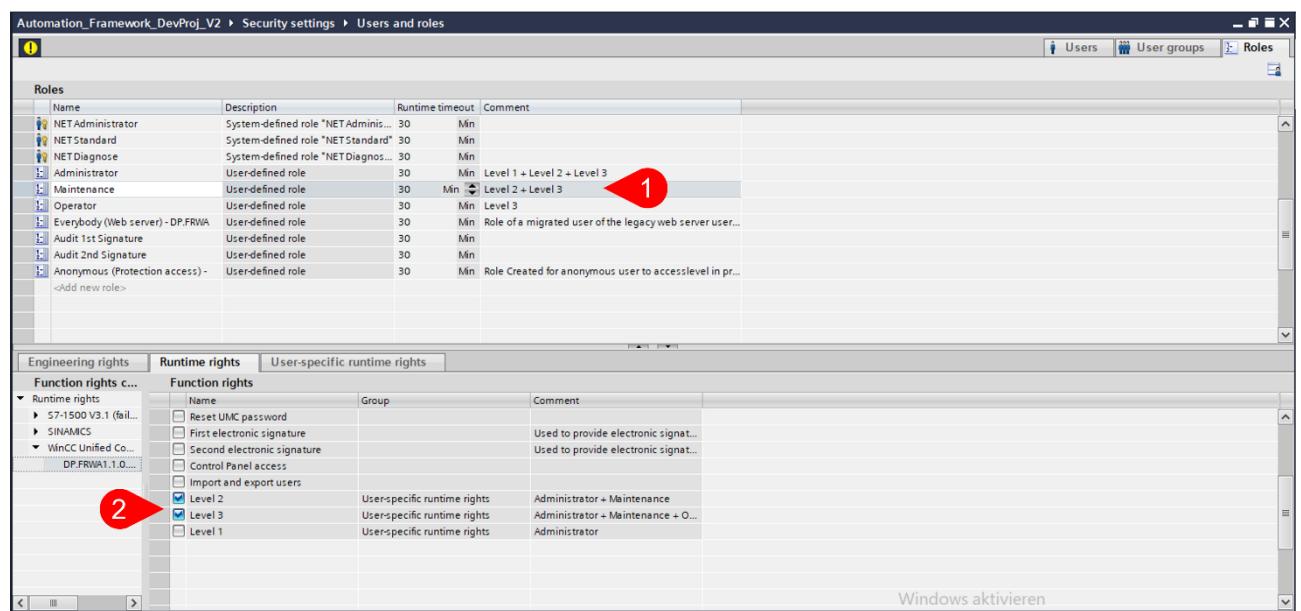


图 17-4 运行系统权限

这些角色可以重命名，也可以通过其他角色扩展列表。要为用户分配角色，请按以下步骤操作：

- 打开“Users”选项卡(1)。
- 选择要为其分配角色的用户(2)。注意不能使用多重选择。
- 在“Assigned roles”部分启用所需的角色(3)。

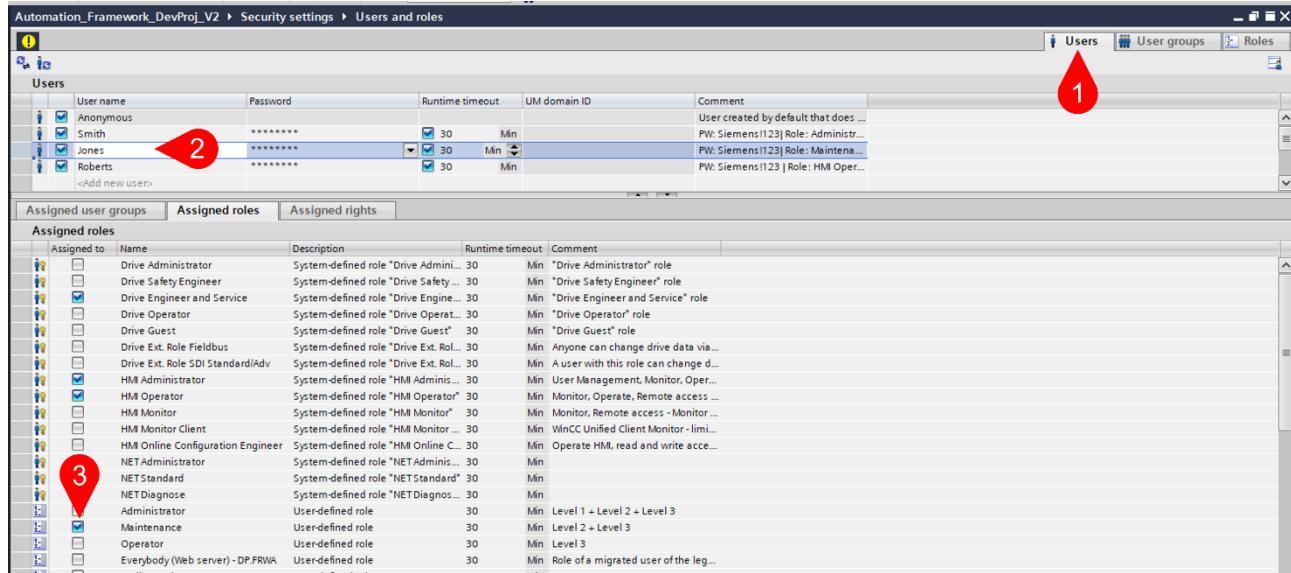


图 17-5 分配角色

组态用户和角色后，需要设置具有相关访问级别的项目和画面。在 AF 项目中，已经实现了一个示例。“Setting”画面(2)应仅对具有“Role”“Level1”的用户开放。因此，需要组态“Security”设置(3)，并且必须将“Authorization”设置为“Level1”。此组态导致了一种行为，即只有角色为“Level1”的用户才能访问 AF HMI 的设置页面。

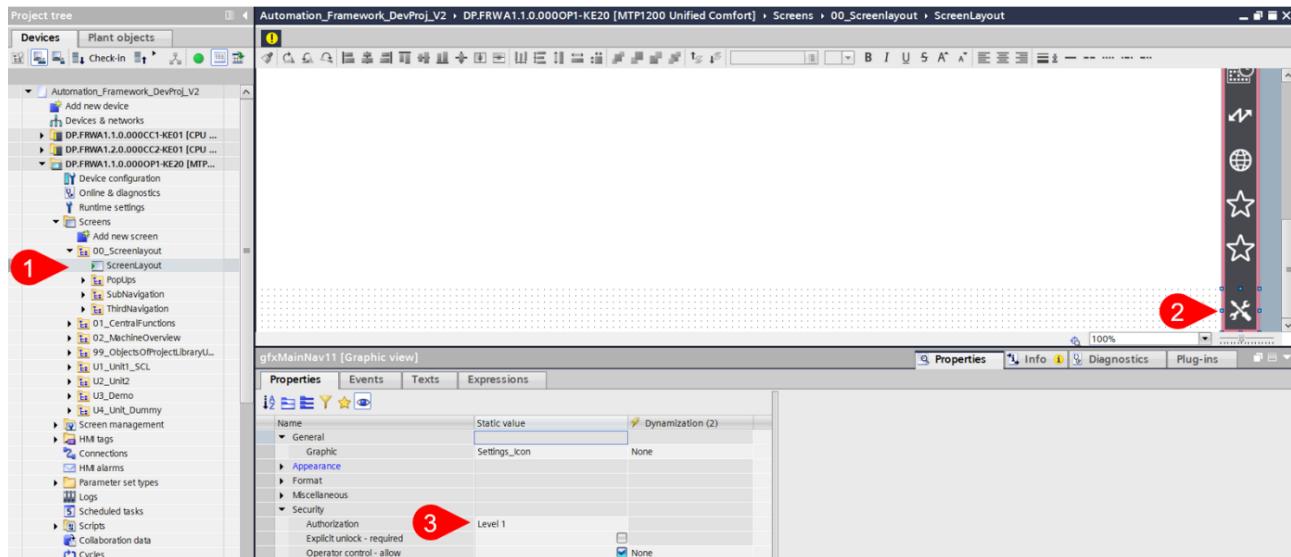


图 17-6 为画面对象分配授权级别

系统定义的角色:

- 1: 驱动管理员
- 2: 驱动安全工程师
- 3: 驱动工程师和服务
- 4: 驱动器操作员
- 5: 驱动访客
- 6: 驱动器外部角色现场总线 – 如果“匿名”用户具有此角色，则任何人都可以通过现场总线协议更改驱动器数据（安全集成以及用户和角色管理的设置除外）。这也适用于“驱动工程师和服务”角色。
- 7: 驱动器外部角色 SDI 标准/高级 - 具有此角色的用户可以通过 SDI 标准面板更改驱动器数据（安全集成以及用户和角色管理的设置除外）。这也适用于“驱动工程师和服务”角色。

- 8: HMI 管理员 - 用户管理、监视、操作、远程访问、远程访问 - 仅监视、Openness Runtime - 读写访问、OPC UA - 读写访问、导入和导出用户、复位 UMC 密码、GraphQL 访问。
- 9: HMI 操作员 - 监视、操作、远程访问 - 仅监视。
- 10: HMI 监视器 - 监视，远程访问 - 仅监视。
- 11: HMI 监视客户端 - WinCC Unified 监视客户端 - 有限访问。
- 12: NET 管理员
- 13: NET 标准
- 14: NET 诊断

用户自定义角色:

- 管理员 – 1 级 + 2 级 + 3 级
- 维护 – 2 级 + 3 级
- 操作员 – 3 级

用户组	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
管理员	X	X	X					X	X			X	X	X	X		
维护			X					X	X							X	
操作员									X								X
匿名																	

表 17-1 运行系统和用户特定权限

18. 运行演示

AF 使用真实硬件进行测试，是一个可运行的程序。但是，某些用例涉及的设备数量多于测试环境中可用的设备数量。因此，一些设备和信号被仿真或简单地表示为数据块中的变量。AF 项目可以使用真实硬件执行，也可以通过仿真执行。为了在仿真模式下运行，需要 PLCSIM 和 PC 运行系统。

这种方法允许在最终部署所有物理设备之前，在受控环境中评估和验证程序的功能。仿真可确保可以测试多个场景和组态，而无需从一开始就需要所有必要的硬件。总之，AF 在测试过程中提供了灵活性和稳健性，确保真实硬件执行和仿真都能提供一致且可靠的结果。

当用户尝试在 PLC (1) 中下载已编译的程序时，会弹出一个窗口，要求输入有权在 PLC 中写入程序的用户登录信息。

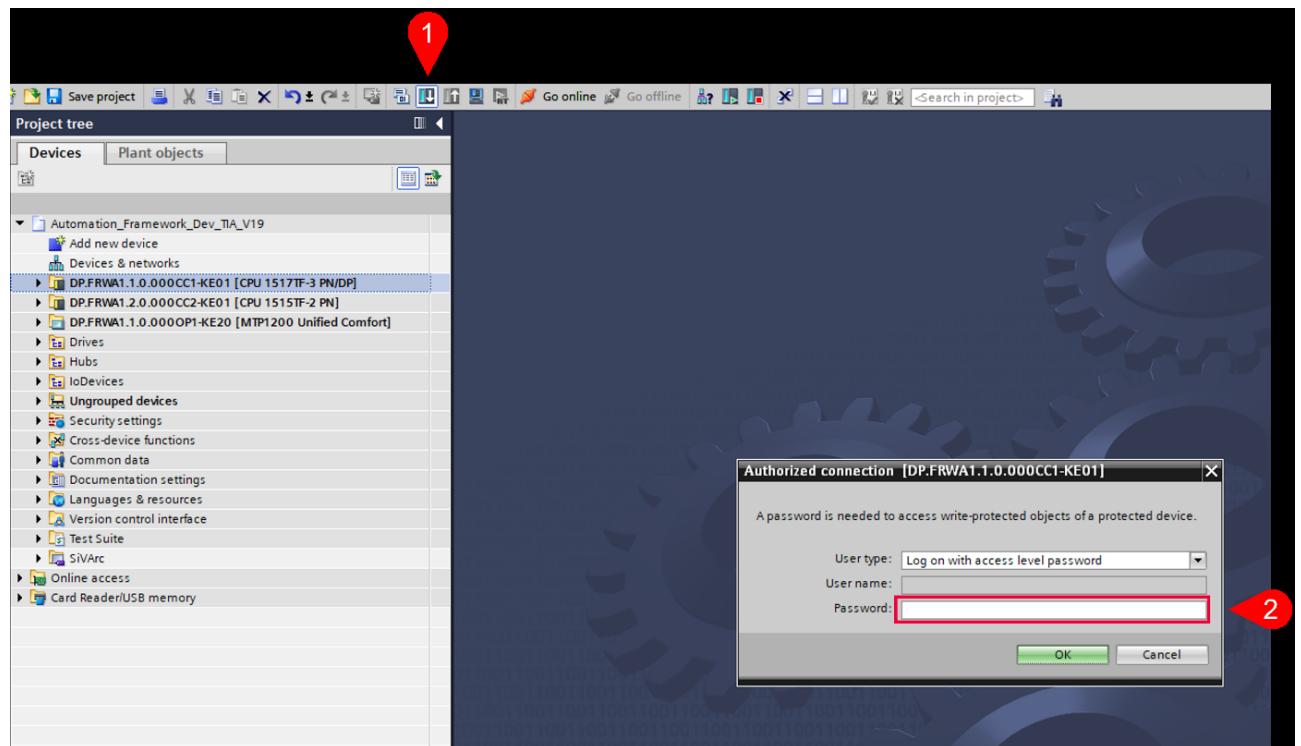


图 18-1：在 PLC 中下载已编译的程序

要选择的用户类型是“Project user”(2)。然后必须填写“User name”和“Password”。用户名和密码信息可在“User and roles”菜单的安全设置中找到（参见第 17 章[用户管理](#)）。出于演示的目的，用户必须具有管理员角色。在这种情况下，登录数据如下：

Automation_Framework_V1.1_20240630 ▶ Security settings ▶ Users and roles

	User name	Password	Runtime timeout	UM ...	Comment
	<input checked="" type="checkbox"/> Anonymous				User created by default that does not n...
	<input checked="" type="checkbox"/> Smith	*****	<input checked="" type="checkbox"/> 30 Min		PW: Siemens!123 Role: Administrator
	<input checked="" type="checkbox"/> Jones	*****	<input checked="" type="checkbox"/> 30 Min		PW: Siemens!123 Role: Maintenance
	<input checked="" type="checkbox"/> Roberts	*****	<input checked="" type="checkbox"/> 30 Min		PW: Siemens!123 Role: HMI Operator
<Add new ...					

Assigned user groups Assigned roles Assigned rights

图 18-2: 具有管理员角色的用户登录数据

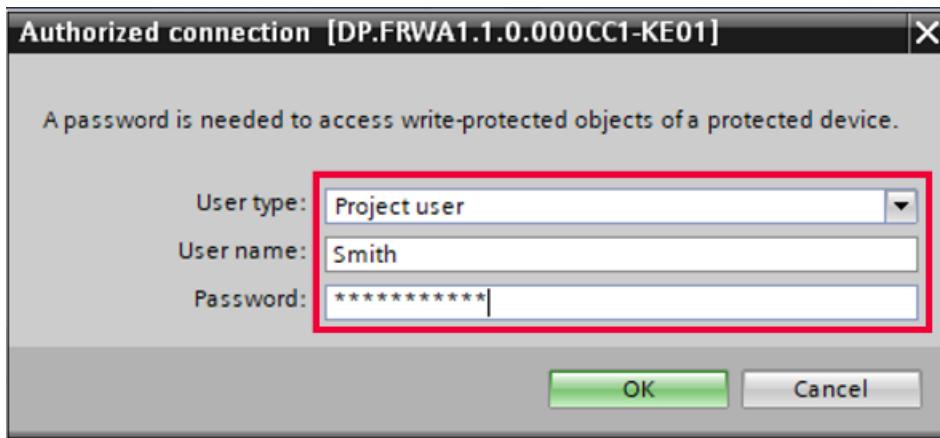


图 18-3: 用户必须填写登录数据才能授权与 PLC 连接

对于人机界面，要运行演示，用户必须仿真面板。仿真运行后，人机界面将无法操作。用户必须使用与 PLC 相同的登录数据和管理员凭证登录。为此，按下/点击“User”字段，弹出登录窗口(1)。现在用户可以填写登录数据(2)，然后就可以操作人机界面了。

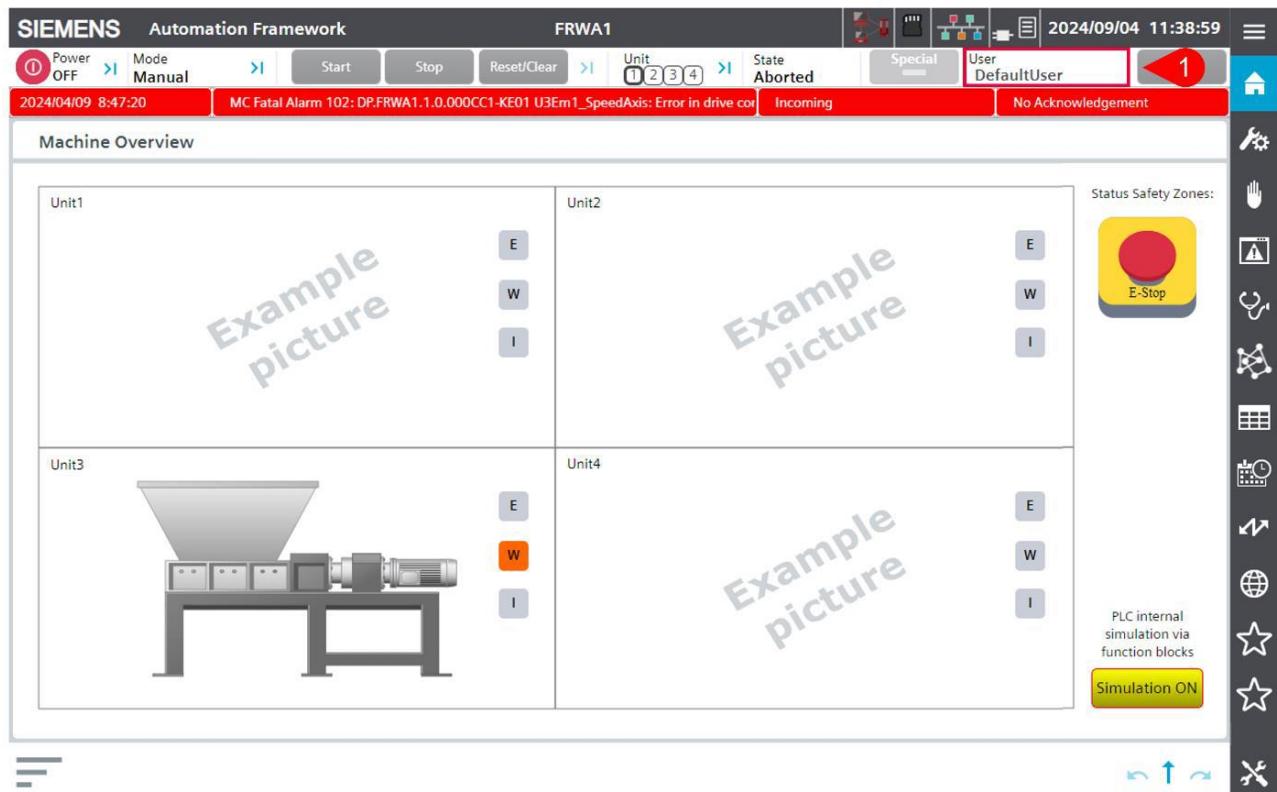


图 18-4: 无法操作的人机界面，用户必须使用管理员凭据登录

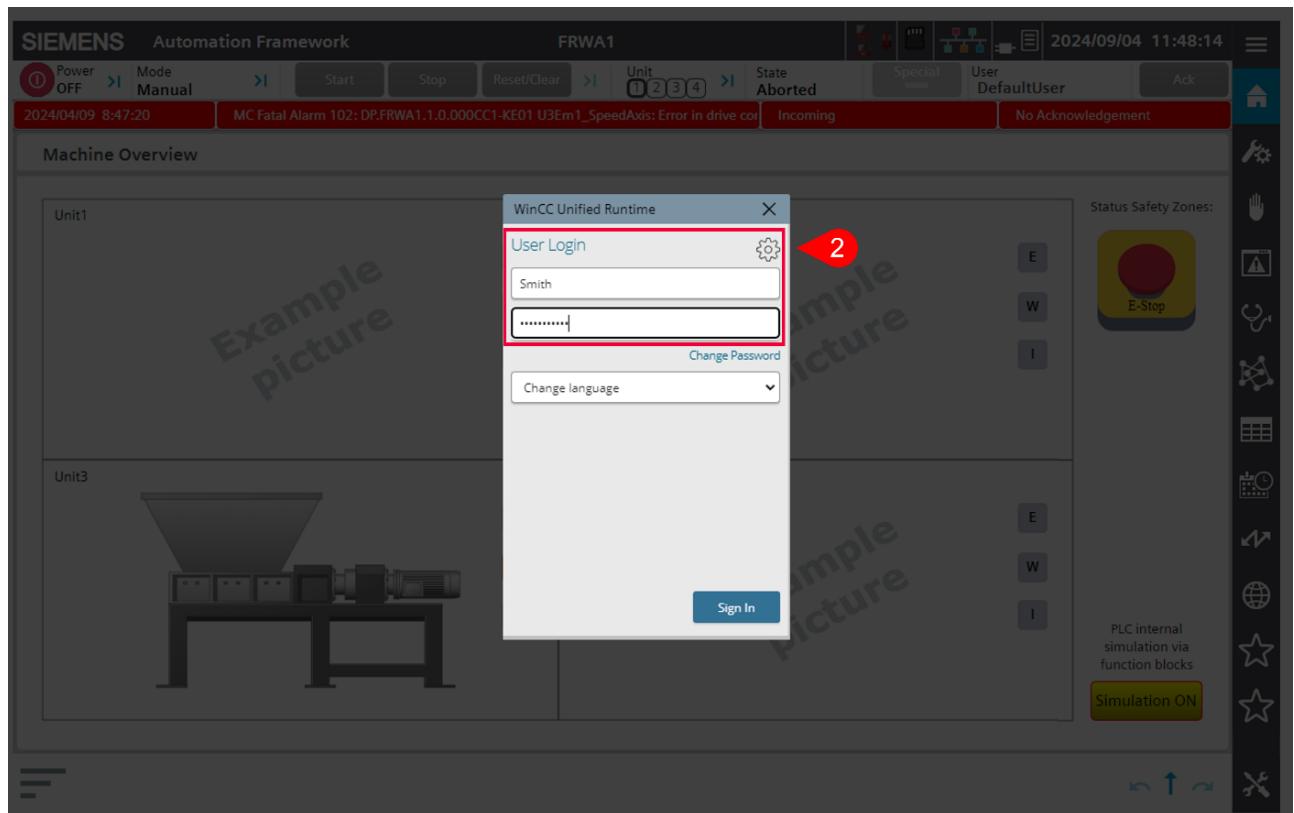


图 18-5: 使用管理员凭证登录人机界面

19. 附录

19.1. 西门子准则

标准化

标准化指南展示如何实现机器和系统的模块化。它提供有关自动化解决方案的结构化和标准化编程的建议和提示。

<https://support.industry.siemens.com/cs/ww/en/view/109756737>

编程指南

SIMATIC S7-1500 控制器及其最新的系统架构应始终与 TIA Portal 一起使用。两者完美协调，并提供高效的编程和组态选项。SIMATIC S7 1200 和 S7-1500 编程指南提供了有关创新、建议和技巧的信息，以便结合使用 TIA Portal 和 SIMATIC S7-1200 和 S7 1500 控制器创建强大的用户程序。

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

安全编程指南

SIMATIC S7-1200 和 S7 1500 的安全编程指南描述了应如何组态故障安全控制器，演示了不同的安全编程方法以及如何优化安全编程的提示。

<https://support.industry.siemens.com/cs/ww/en/view/109750255>

编程风格指南

PLC 程序应尽可能具有可读性和结构性，以便每个软件开发人员，调试人员和维护工程师都可以阅读和理解代码。

如果每个开发人员都按照自己的理念来完成任务，那么产生的代码就不可避免地只能由各自的创建者来解释。

通过《SIMATIC S7-1200 和 S7-1500 编程风格指南》，西门子为创建统一的程序代码提供了一些帮助，即使在多个开发人员使用同一应用程序的情况下，该代码也是可维护和可重用的。

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

WinCC Unified 的工程组态指南

Unified Panel 和 PC-RT 设备提供最新的技术，如 HTML5、JavaScript 和可缩放矢量图形(SVG)。为了有效地使用它们，应该遵循一些工程组态建议。

如果开始一个新项目，但如果已经有一个现有项目，那么所呈现的实现可能会有所帮助。

<https://support.industry.siemens.com/cs/ww/en/view/109827603>

Test Suite Advanced：检查 TIA Portal 项目是否符合编程风格指南

该应用示例提供了一个规则集，用于使用 TIA Portal Test Suite Advanced 检查风格指南。一个演示项目展示了风格指南检查的工作原理。

<https://support.industry.siemens.com/cs/ww/en/view/109779806>

TIA Portal 中的库处理指南

《TIA Portal 中的库处理指南》描述了在 TIA Portal 中使用典型化库元素创建企业库。使用此类类型具有一些优点，例如，所有实例的中央更新功能、系统支持的版本管理和库元素的更改跟踪。

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

TIA Portal 中的多用户工程组态

借助 TIA Portal 中的多用户工程组态，可以同时与多个用户在一个项目上进行工作。通过在多用户项目中并行处理不同的对象，可以显著缩短项目规划和调试时间。

<https://support.industry.siemens.com/cs/ww/en/view/109740141>

19.2. TIA Portal 选件

TIA Portal Test Suite

TIA Portal Test Suite 能够定义一个规则集，该规则集包含多个规则，根据这些规则验证函数块、函数、组织块、各种全局 DB/背景数据块、PLC 变量和用户自定义的数据类型。

<https://support.industry.siemens.com/cs/ww/en/view/109825229>

SIMATIC Visualization Architect (SiVArc)

借助 SIMATIC Visualization Architect (SiVArc)，TIA Portal 提供了一个选件，可以借助定义控制程序和可视化之间分配的规则来生成可视化。使用此选件可以节省组态时间并防止错误，尤其是在重复任务中。

<https://support.industry.siemens.com/cs/ww/en/view/109826234>

19.3. 用于共享任务的软件一目了然

SIMATIC 软件提供的强大工具将为您节省宝贵的时间，使您能够专注于核心自动化任务。以下链接显示了该软件的概述：

<https://new.siemens.com/global/en/products/automation/industry-software/automation-software/software-for-shared-tasks.html>

TIA 选型工具

对于应用，我们提供 TIA 选型工具，为所有项目规划人员、初学者和专家提供支持。无需详细的投资组合知识。TIA 选型工具可作为免费桌面版本或云版本下载。

<http://www.siemens.com/TIA-Selection-Tool>

直接从 <https://mall.industry.siemens.com/tstcloud/#/Start> 开始

Sinetplan

西门子网络规划器为基于 PROFINET 的自动化系统规划人员提供支持，并支持对系统网络进行专业和主动的仿真。

<https://support.industry.siemens.com/cs/ww/en/view/109763136>

PRONETA

PRONETA Professional 支持自动化系统的操作员对所用组件进行自动数据采集，并在 PROFINET 网络中使用全面的诊断功能。

专业版 <https://support.industry.siemens.com/cs/ww/en/view/109781283>

基本版 <https://support.industry.siemens.com/cs/ww/en/view/67460624>

SIMATIC Automation Tool

SIMATIC Automation Tool V5.0 支持与 TIA Portal V18 相关的最新固件版本，并提供附加功能以及额外的 SDK 变体。

以下是此版本中的新增功能：

- 支持与 TIA Portal V18 版本相关的其他新设备和最新固件版本，如设备目录中所示
- 硬件组态与实际硬件的比较
- F-CPU 的 F-Signature 设备表显示
- 对“插入设备”对话框的改进
- 带有设备和操作信息的状态栏
- 用于确定设备是否已连接的快速 ping 操作。

<https://support.industry.siemens.com/cs/ww/en/view/109816217>

Siemens OPC UA 建模编辑器 (SiOME)

通过免费的“Siemens OPC UA Modelling Editor”(SiOME)工具，可以创建编辑器，用于在 SIMATIC PLC/SINUMERIK 上定义用户自己的 OPC UA 信息模型或映射现有的配套规范。使用该工具，用户可以导入和编辑 XML 文件形式的信息模型，或生成和导出个性化模型。

<https://support.industry.siemens.com/cs/ww/en/view/109755133>

19.4. 驱动器的应用

SINAMICS SDC: Serial Drive Commissioner

Openness 应用程序 SINAMICS SDC (Serial Drive Commissioner) 提供复制、更新和下载多个驱动器的功能。支持 SINAMICS G 产品线的设备和 SINAMICS S210 驱动器。

<https://support.industry.siemens.com/cs/ww/en/view/109774753>

在多个驱动器中编辑参数 Add-In

TIA Portal V16 通过将现有组态从一个驱动器对象复制到其他驱动器对象，从而方便相似配置的驱动器调试。参数值可以离线或在线写入。

<https://support.industry.siemens.com/cs/ww/en/view/109777633>

SIMATIC S7-1500 / S7-1500T: 标准应用轴控制

功能块用于对 SIMATIC S7-1500 或 S7-1500T 的轴（工艺对象）的基本运动控制功能进行简单的集中控制。通过该标准应用程序对每个轴进行集中查看，可实现轻松编程、快速调试和直接测试应用程序。

建议使用“LAxisCtrl”库。

<https://support.industry.siemens.com/cs/ww/en/view/109749348>

SIMATIC - 故障安全库 LDrvSafe，用于控制 SINAMICS 驱动器系列的安全集成功能

该库包括故障安全 SIMATIC S7 块，可与 S7-1200F、S7-1500F、故障安全开放/软件控制器和通过 PROFIsafe 连接 SINAMICS 驱动器配合使用，实现各种安全应用。

<https://support.industry.siemens.com/cs/ww/en/view/109485794>

使用 SIMATIC Energy Management (S7-EE Monitor) 对机器进行能效评估和介质分析

根据 VDMA34179 或 ISO14955 对机器进行标准化效率评估的概念和应用示例允许评估机器（所有介质）的能源效率。

<https://support.industry.siemens.com/cs/ww/en/view/109753230>

20. 附录

20.1. 服务和支持

SiePortal

产品选择、购买和支持的集成平台，以及工业商城和在线支持的连接。SiePortal主页取代了以前的Industry Mall和在线支持门户（SIOS）主页，并将它们组合在一起。

- 产品与服务

在产品和服务中，您可以在购物中心目录中找到我们之前提供的所有产品。

- 支持

在支持中，您可以找到所有有助于解决我们产品技术问题的信息。

- mySieportal

mySieportal收集您的所有个人数据和流程，从您的帐户到当前订单、服务请求等。登录后，您才能看到全部功能。

您可以通过以下地址访问SiePortal: sieportal.siemens.com

技术支持

西门子工业技术支持部门为您提供快速而有力的支持，包括从基本支持到单个支持合同的众多量身定制的服务，以解决所有技术问题。请通过 Web 表单向技术支持发送查询: siemens.com/SupportRequest

SITRAIN – Digital Industry Academy 数字化工业学院

我们为您提供全球可用的行业培训课程，提供实践经验、创新的学习方法和根据客户特定需求量身定制的概念。

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

Industry Online Support app

无论您身在何处，都可以通过“行业在线支持”应用程序获得最佳支持。该应用程序适用于 iOS 和 Android:



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

20.2. 文献链接

序号 主题

1	Siemens Industry Online Support https://support.industry.siemens.com
2	Link to this entry page of this application example https://support.industry.siemens.com/cs/ww/en/view/109817223
3	Siemens OPC UA 建模编辑器 (SiOME) https://support.industry.siemens.com/cs/ww/en/view/109755133
4	用于生成文档的 TIA Portal 插件 Code2Docu https://support.industry.siemens.com/cs/ww/en/view/109809007
5	SIMATIC OMAC PackML V2022 模式和状态管理和机器数据接口 https://support.industry.siemens.com/cs/ww/en/view/109821198
6	适用于制造用例的西门子信息模型套件 https://support.industry.siemens.com/cs/ww/en/view/109814835

表 20-1 链接和文献

20.3. 变更文档

Version	Date	Modification
V1.0	2023/12/22	First version
V1.1	2024/06/28	N.a., as version V1.0 was not used.
V1.2		<ul style="list-style-type: none">● Hardware configuration:<ul style="list-style-type: none">- ET200SP (-KE60) replaced by ET200SP with fail-safe I/O modules- ET200SP (-KE50) integrated into PROFINET IRT line topology- Additional G120, S120 and S200 drives added for tests and simulation- (010CV1-DU3, 020CV1-DU3, 020CV1-DU4, 050CV1-DU1, 060CV1-DU1, 060CV1-DU2)- Gaps between I/O addresses reduced to improve simulation performance- Standardized I/O tag names according to module designation- I/O tags organized in the tag table of the corresponding equipment module● LPD library for PROFIdrive telegrams added● PLC data types from LGF library for PLCopen diagnostics moved to LAF library● Safety Unit:<ul style="list-style-type: none">- Data exchange between standard program and safety program: Update to the use of the Safety Unit, and new safety programming guidelines.- Updated structure of the safety program- New fail-safe I/O modules included into the safety program- Updated workflow for simulation of fail-safe I/Os● LBC 3.0 update● TmAspiration:<ul style="list-style-type: none">- Changed reference designator from string to WString for FB and UDT- Added missing comments to interface- Removed unused "SinaSpeedScaleOut"- Removed not needed HMI/OPC UA accessibility flag● CentralFunctions:<ul style="list-style-type: none">- Simulation functionality added. Central management of simulation tags.- Manual Operation Line faceplate fixed triggerDataSet call because of Tia V19 Upd3. Also fixed "unit" property not being shown.● Manual Operation Lines:<ul style="list-style-type: none">- Block documentation correction.- Default tag connected for input "screenData" removed● LAF SelectJob:<ul style="list-style-type: none">- Rework of the Code- Reset behaviour of the "jobCompleted" Bit adapted- Input, Output and InOut names changed● LAF SCLControl: Initialization behaviour adjusted● Call Environments:<ul style="list-style-type: none">- "statReferenceDesignator" deleted and connections of the variable changed to the combined Reference Designator within the DB "Config"- Control Module calls separated. Sensors at the beginning of the EM logic, Actuators are called after processing the sequence logic● Sequence blocks:<ul style="list-style-type: none">- Stopped step has been deleted and the sequences only have an Immediate Stopped step that is activated when the command "immediateStop" of the sequence interface is triggered.- "LAF_UpdateJobHistory" call added to the Status update section