西门子工业
在线支持

应用示例

WinCC Unified 组态指南

SIMATIC WinCC Unified V18 / V19/ V20/ V21

SIEMENS

法律信息

使用应用示例

应用示例以文本、图形和/或软件模块的形式，通过多个组件之间的交互来说明自动化任务的解决方案。应用示例是西门子股份公司和/或西门子股份公司的子公司（"西门子"）提供的免费服务。这些示例不具有约束力，也不保证配置和设备的完整性或功能性。应用示例仅为典型任务提供帮助，并不构成客户特定的解决方案。您自己有责任按照适用的规定正确、安全地操作产品，还必须检查相应应用示例的功能，并根据您的系统进行定制。

西门子授予您非排他性、不可再授权和不可转让的权利，允许受过技术培训的人员使用应用示例。对应用示例的任何更改均由您负责。与第三方共享应用示例或复制应用示例或应用示例节选仅允许与您自己的产品结合使用。应用示例无需经过收费产品的常规测试和质量检验；可能存在功能和性能缺陷以及错误。您有责任在使用这些示例时，确保可能发生的故障不会导致财产损失或人身伤害。

免责声明

西门子不承担任何法律责任，包括但不限于应用示例的可用性、可能性、完整性和无缺陷性，以及相关信息、配置和性能数据和由此造成的任何损害。这不适用于强制责任的情况，例如根据《德国产品责任法》，或故意、重大过失或过失造成生命损失、人身伤害或健康损害的情况，不遵守担保、欺诈性不披露缺陷或过失违反重要合同义务的情况。但是，因违反重要合同义务而产生的损害赔偿要求应仅限于协议类型中典型的可预见损害，除非责任产生于故意或重大过失，或基于生命损失、身体伤害或健康损害。上述规定并不意味着举证责任发生任何不利于您的变化。对于第三方在此方面提出的现有或未来索赔，您应向西门子作出赔偿，但西门子必须承担责任的情况除外。通过使用应用示例，您承认西门子不对所述责任条款之外的任何损害承担责任。

其他信息

西门子保留随时更改应用示例的权利，恕不另行通知。如果应用示例中的建议与西门子的其他出版物（如目录）不一致，则以其他文件的内容为准。
西门子使用条款 (https://www.siemens.com/global/en/general/terms-of-use.html) 同样适用。

安全信息

西门子提供具有工业安全功能的产品和解决方案，支持工厂、系统、机器和网络的安全运行。
为了保护工厂、系统、机器和网络免受网络威胁，有必要实施并持续维护全面、先进的工业安全理念。西门子的产品和解决方案就是这种理念的一个组成部分。
客户有责任防止未经授权访问其工厂、系统、机器和网络。这些系统、机器和组件只能在必要时连接到企业网络或互联网，并且必须采取适当的安全措施（如防火墙和/或网络分隔）。
有关可能实施的工业安全措施的更多信息，请访问
https://www.siemens.com/industrialsecurity。
西门子的产品和解决方案经过不断开发，安全性更高。西门子强烈建议在产品更新可用时立即应用更新，并使用最新的产品版本。使用不再支持的产品版本以及未应用最新更新可能会增加客户面临网络威胁的风险。
要随时了解产品更新信息，请在以下链接订阅西门子工业安全 RSS 源。
https://www.siemens.com/cert 。

目录

1. 引言

1.1. 概述

SIMATIC WinCC Unified 是西门子在自动化环境中全新开发的可视化系统。SIMATIC WinCC Unified 系统由 SIMATIC WinCC Unified 可视化软件、新型 SIMATIC HMI Unified 精智面板和Unified 精简面板组成。

WinCC Unified 面板扩展了基于 SIMATIC 高级面板的 HMI 产品范围，是 SIMATIC HMI 精智面板的后续设备。除了新的硬件外，还有许多创新的新功能，会深刻改变您的使用方式。本文档将介绍几个主题，让您更好地了解如何使用WinCC Unified 面板和PC Runtime 中的新功能和行为。我们将讨论其优势，并指导您完成更改流程的步骤。

本文档内容涉及整个 Unified HMI 产品组合，除非另有说明涉及到特定设备信息。

1.2. 使用的组件

本应用示例使用以下硬件和软件组件创建：

组件	数量	文章编号	备注
WinCC Unified 工程 V18	1	6AV215.-...01-8A.5	TIA 博途中的工程
WinCC Unified 工程 V19	1	6AV215.-...02-3.A5	TIA 博途中的工程
WinCC Unified 工程 V20	1	6AV2151-.....-4AA5	TIA 博途中的工程
SIMATIC HMI Unified PC Runtime V18	1	6AV2154-..B01-8.A0	运行系统
SIMATIC HMI Unified PC Runtime V19	1	6AV2154-..B02-3.A0	运行系统
SIMATIC HMI Unified PC Runtime V20	1	6AV2154-.....-4AA0	运行系统
SIMATIC HMI Unified PC Runtime V21	1	6AV2154-.....-5AA0	运行系统
SIMATIC HMI Unified 精智面板 MTP1200	1	6AV2128-3MB06-0AX1 或者，也可以使用其他 SIMATIC HMI Unified 精智面板	

您可以从西门子工业商城购买这些组件。

注意 请注意，本文档中展示或说明的所有功能仅全面适用于 WinCC Unified V21。部分未额外标注的内容也可适用于WinCC Unified V18、 V19和V20包括更新。

本应用示例涉及以下组件：

组件	文件名	备注
手册	109827603_WinCC_Unified_engineering_guideline_DOC_V2_en.pdf	源英文文件

1.3. 所用符号说明

下表列出了所使用的符号，这些符号表示所描述的建议适用于哪个版本。如果没有使用符号，则表示内容对版本没有限制。

符号	Unified 版本
V18	仅与 V18 有关
V19	仅与 V19 有关
V19 Upd2	仅与 V19 Upd2 有关
V20	仅与 V20 有关
V21 Upd3	仅与 V20 Upd3 有关
V21	仅与 V21 有关

2 基本信息

2.1. 使用《组态指南》的原因

Unified 面板和PC-RT 设备采用了最新技术，如 HTML5、JavaScript 和可缩放矢量图形 (SVG)。这使得它们比以前的精智面板和 WinCC RT Advanced 系统功能更强大，能够提供更丰富的功能，让您充分发挥系统的潜力。为了更有效地使用这些系统，本文件提供了一些工程设计建议。如果您要开始一个新项目，或者您已经有了一个现有项目，本文档中介绍的实施方案会对您有所帮助。

在创建新项目时，建议首先阅读组态指南。它将提供以最有效的方式使用设备资源的知识。

此外，如果已有 Unified HMI 项目，本指南还有助于进一步改进项目的实施。本指南还有一个明确的章节，介绍如何分析项目，以找到易于改进的地方。

有关Java Script 代码的一般优化、脚本样式指南和 HMI 布局，请参阅附加文档：样式检查器、 HMI 模板套件

、脚本的提示和技巧

所有工具也可在第 5.2 节的链接和文献中找到。

2.2. 选择正确的硬件

为确保功能范围准确且性能最佳，选择合适的硬件至关重要。在Unified环境中，您有三种主要选项：

Unified PC Station

Unified Comfort Panel

Unified Basic Panel

面板和IPC在尺寸和资源方面存在差异，所以首先需要在这三者之间进行选择。

> 注意 系统限制
> 为确保设备符合系统限制，请检查各自的系统限制 >链接

2.2.1. 自动化框架

在考虑不同可视化设备的应用时，最简便的划分方式是按应用范围进行区分。根据应用场景的不同，PC 工作站可能更适合工厂环境，而面板则更适用于直接安装在机器上的应用。除按HMI应用领域划分外，“自动化框架”与“基础自动化框架”还存在差异——它们在需要实现的自动化复杂度方面有所不同。
当仅需覆盖基础应用场景时——无论是单台设备还是单一工艺流程——WinCC Unified 基础面板配合小型控 制器即可构成恰当且充分的硬件配置。对于涉及多台设备或多工艺流程的大型项目，自动化框架为PLC和 HMI编程提供了专业架构，其全面功能可完美适配Unified Comfort Panel及S7-1500 PLC系列产品。

请注意，这些只是粗略的指导方针，必须针对每个特定应用单独决定！下图显示了按区域和应用复杂性的两种区分， 但仅指HMI设备，而不指PLC。

图 2-1 应用范围–AF

基础和高级应用
自动化框架专为机械制造商量身打造，为PLC和HMI程序提供核心架构，确保从更高层级IT系统控制和监控设备状态时具备标准化接口。其区别在于：

UBP ：配备小型控制器，适用于覆盖单台设备的基础自动化需求
UCP：针对覆盖多台设备的高级自动化需求

为进一步说明画面可容纳的元素数量，AF和AF基础版提供了基于以下架构的示例画面：单元 > 设备模块 > 控制模块 这些建议一方面确保操作员不会因信息过载而疲于应对，能够轻松直观地操作人机界面；另一方面则确保操作始终在 系统限制和设备资源范围内进行。

图 2-2 自动化框架单元概念

自动化框架
要进一步了解自动化框架，请查看以下链接：
[自动化框架]
[自动化框架基础]

2.3. 新配置的工作流程

当您开始创建全新的配置，甚至只是创建新的画面时，就是您以最有效的方式使用新的 WinCC Unified 功能的最佳时机。在这种情况下，视图一词指的是 HMI 的整个显示画面，因此也指画面窗口排列中多个画面的组合。创建新视图时， 第一步是预先收集所有需要显示的内容。根据其可重用性，需要将内容分类为单个内容或将在多个画面中使用的内容。内容最有效的处理方式在下面的流程图中作了说明。

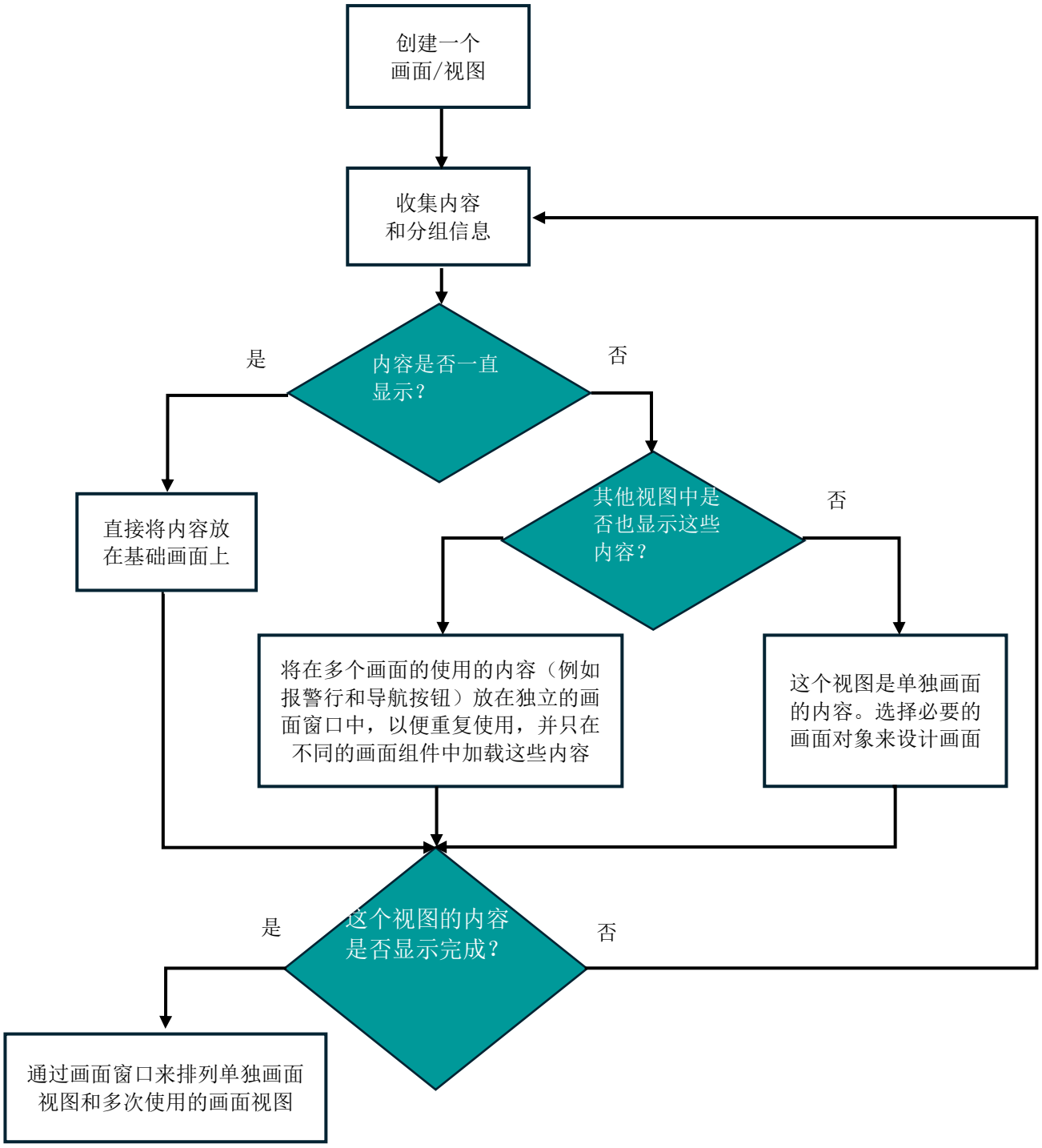


图2-3流程图创建带画面的视图

这种配置的结果可能是如下图所示的画面布局。所有单独的画面窗口都用橙色框突出显示。

如需项目结构指导，请考虑使用 HMI 模板套件。该模板有助于创建和组织显示区域，将其划分为具有多功能画面导航功能的不同画面段（见图 2-2）。它优先考虑性能和设计方面，确保直观的操作概念。

注意 有关设置和使用的详细说明，请参阅应用程序示例 HMI design with the HMI Template Suite

图2-4 使用多个画面窗口的视图示例

一旦确定了哪些元素要放在哪个画面上，就可以继续处理各个画面了。大多数画面项目都需要动态属性，或使用事件来触发系统功能或脚本。为实现这些动态功能，以下方案将向您展示哪种实现方式最适合您的应用程序。可以根据所需的功能，从工具箱中选择一个合适的元素，并将其拖拽到特定画面上。

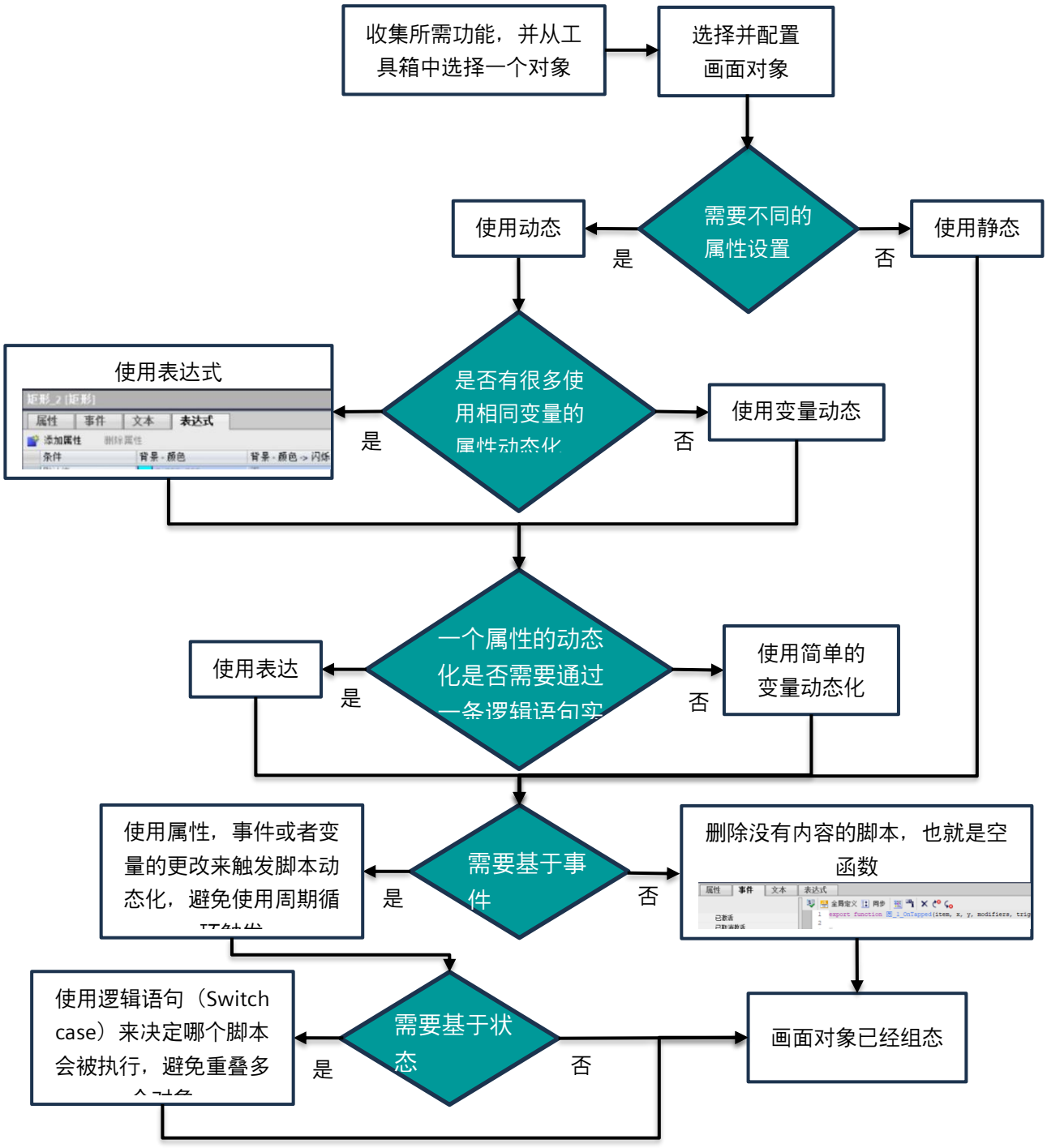


图2-5 配置新画面项目流程图

注意 一般来说，变量动态化是最推荐的动态化类型。如果需要更复杂的逻辑，请使用表达式。尽量避免脚本动态化。不过，如果脚本逻辑可以替换多个重叠的画面项目，则应使用脚本。例如，一个按钮可以显示和/或隐藏其他元素：使用一个按钮，并通过脚本决定执行哪种方法（显示或隐藏）（见 2.2.1章节）。

下一节将通过一个简单的用例解释用逻辑脚本代替多个对象的做法。

2.3.1. 使用案例：状态相关脚本：显示/隐藏画面内容

WinCC Unified 中的脚本现在为配置画面元素提供了更多选项，使其更加灵活。如以下用例所述，单个按钮可根据当前状态触发不同的操作。

说明

事件背后的具体操作或脚本通常取决于当前状态。在本用例中，有一个按钮来显示和隐藏特定的画面内容（此处为矩形），这取决于该画面内容的当前可见性。一种解决方案是使用两个不同的按钮，分别执行显示和隐藏操作，并根据当前状态改变这两个按钮的可见性。第二种解决方案是，只使用一个按钮，并通过脚本（Switch...Case）决定应执行的操作。

解决方案

使用单个按钮来完成这项任务，并执行脚本逻辑来决定是否需要将内容的可见性设置为可见。直接使用相同的变量（此处为"ContentVisible"）来实现。

- 当前状态的指示器
- 内容可见性的动态化
- 按钮文本的动态化

由于这个用例非常简单，脚本逻辑中"ContentVisible"状态变量的转换也可以由系统函数通过按钮点击事件来完成。但为了演示更复杂用例的实现过程，截图显示了带开关用例的实现过程。

要更改不同状态下的按钮文本，可通过文本列表和状态变量进行动态设置。

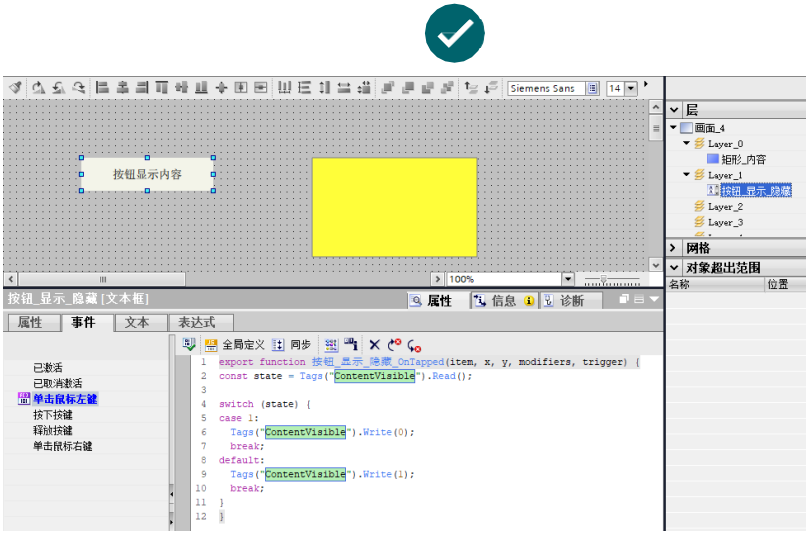


图2-6 显示-隐藏按钮用例脚本逻辑

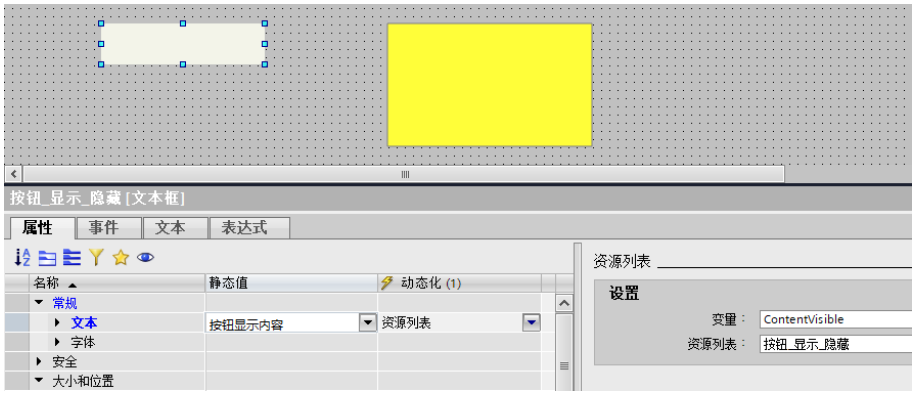


图 2-7 按钮显示-隐藏用例按钮文本的动态化

在以下不推荐的解决方案中，使用了两个叠加按钮，一个用于将矩形的可见性设置为"true"，另一个用于将可见性设置为"false"。为了使按钮能在正确的时间被访问，它们的可见性也会随矩形的可见性状态而动态变化。对于这种需要根据状态进行操作的用例，前面所展示的脚本解决方案比多个叠加画面对象更好，可以最大限度地减少每个画面的对象数量。

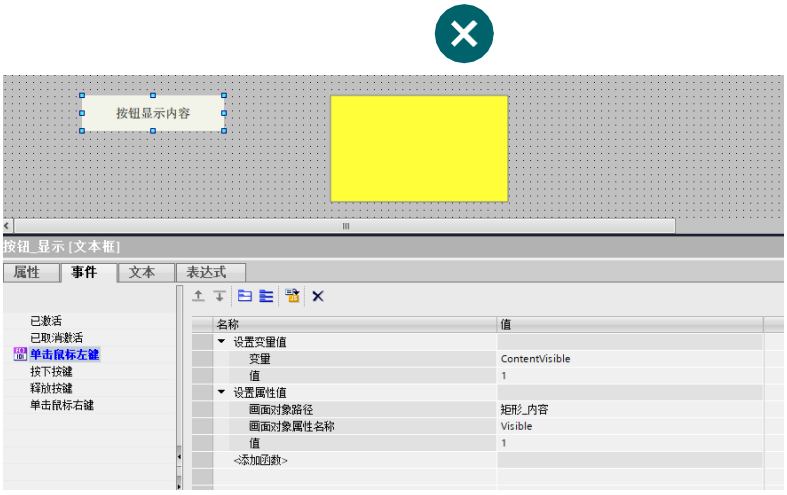


图2-8 用于显示-隐藏-任务的两个重叠按钮

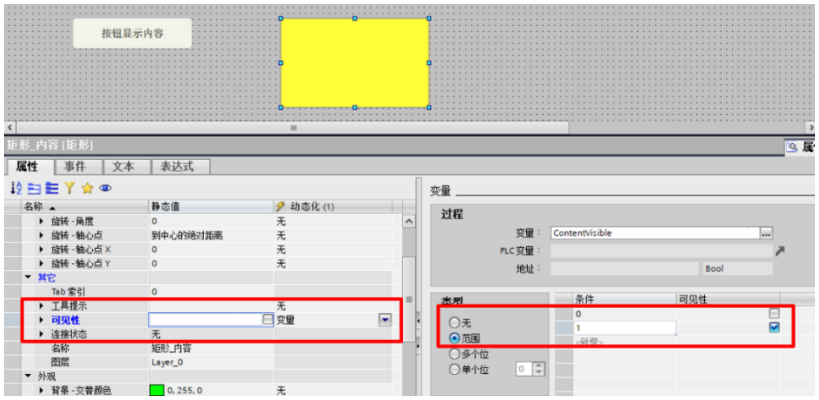


图2-9 两个按钮可见性的动态化

3.1.1.2 用例：不执行按下和释放事件的按钮中解释了用例中将文本框作为按钮的原因。

2.4. 有关画面切换的部分

在 Unified 以及其他可视化系统中，画面更改在各种工厂画面的表现和用户体验中起着核心作用。快速的画面切换会让用户感觉到项目元素的高效性，而缓慢或长时间的画面更改则会让用户感觉不舒服。通过一些小技巧，您可以更好地进行画面切换。

要充分利用组态步骤，就必须了解如何以最佳方式实现这些步骤。在深入了解 "正确 "的实现方式之前，熟悉画面在 Runtime 上的加载过程也很重要。

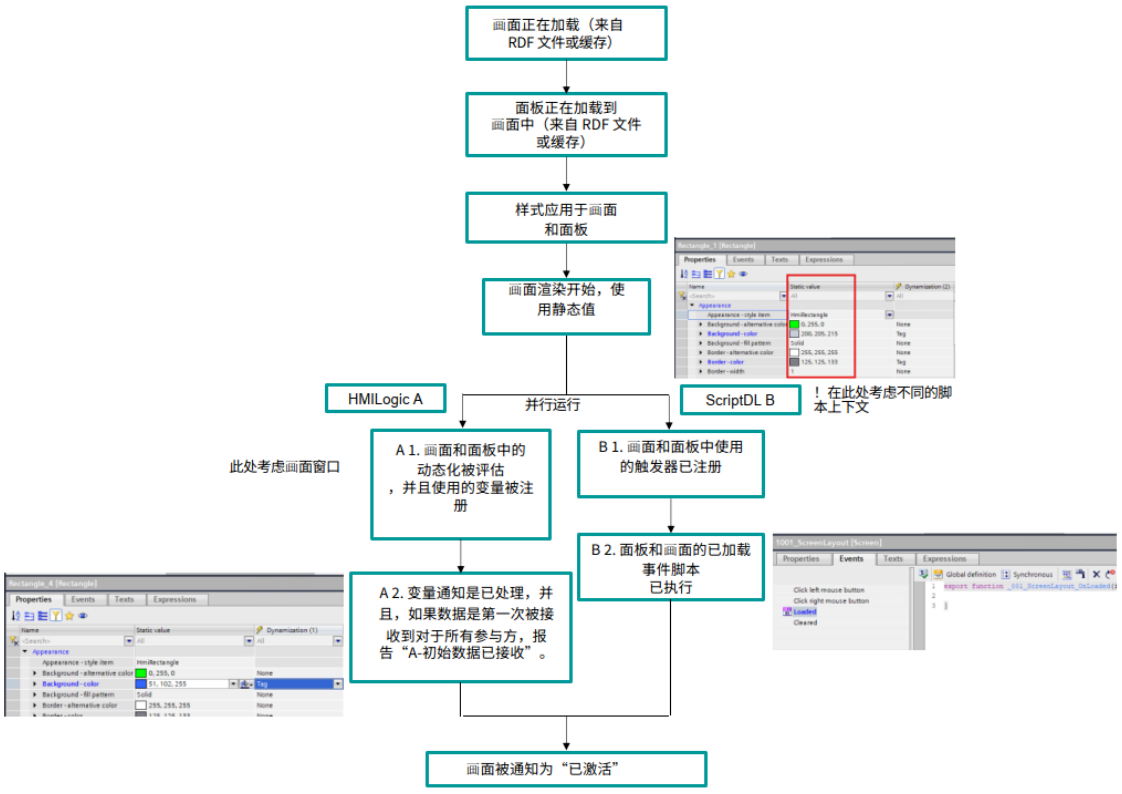


图 2-8 画面加载过程

单个画面的画面加载过程大多是同步分段进行的，这意味着一件事情会依次发生。由于布局通常是由嵌套的画面和面板组成，因此必须知道，每个画面窗口和面板实例都有自己的解耦循环。

在画面切换开始时，甚至在加载新画面之前，前一个画面就会被清除。

完成该步骤后，将构建基本画面，并以静态值加载所有属性（即使这些属性是动态的，例如与变量相关联）。画面窗口中的所有嵌套面板和画面都会独立的加载各自事件。

接下来执行画面 "已加载 "事件，该事件也可在工程组态中自定义。

变量注册发生在画面初始加载之后。变量注册完成后，所有的动态化都会被考虑在内，动态化区域内有连接变量的画面项目属性可以从静态值重新变为动态化变量的值。

如果与任何动态化或任何属性相关联的变量在此上下文中再次更改，画面对象的属性也会再次更改。通过这个执行顺序，在画面加载过程中，属性的更改事件可以执行两次。在下图（图 2-9）中，更改事件的第二次调用以橙色标出。

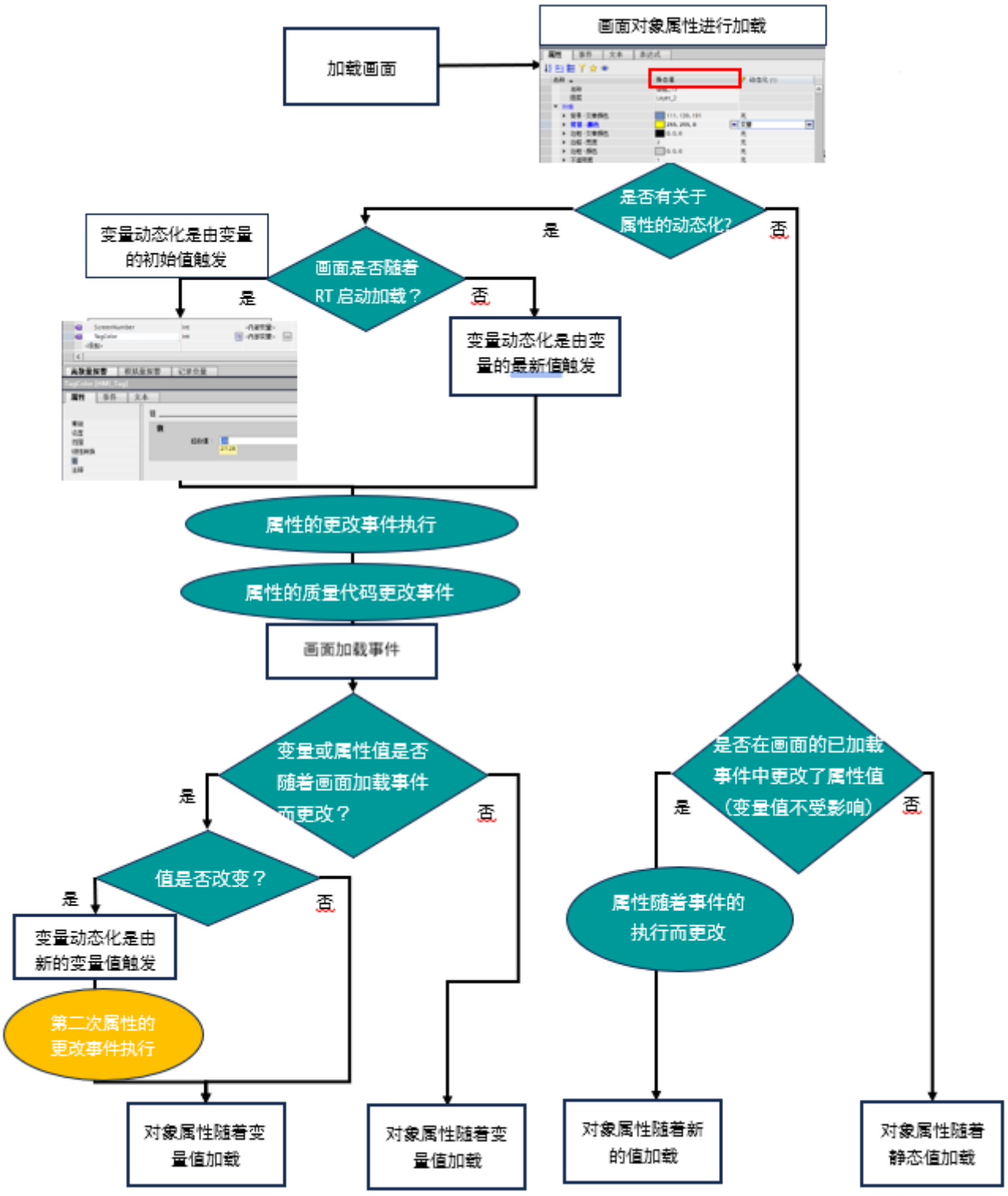


图 2-9　在画面加载过程中调用画面项目属性更改事件

下图显示了使用 PLC 变量进行动态化时的不同行为。重要的一点是 PLC 采集周期时间和质量代码事件的第二次执行时间。有关采集周期的更多信息，请参阅 “采集周期 ”部分。

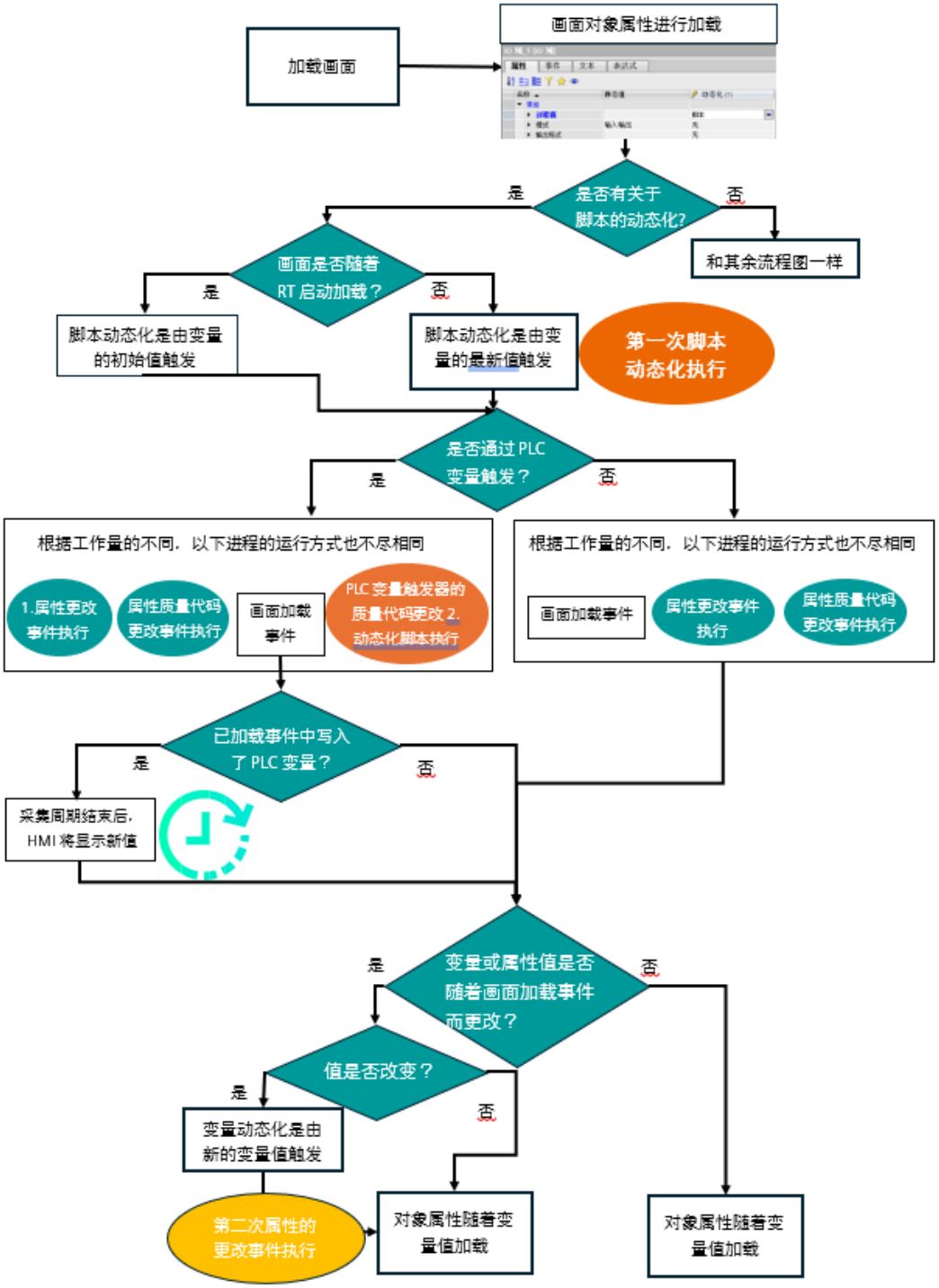


图 2-10 使用 PLC 变量进行动态设置

单个画面的加载过程不仅在一定程度上被划分为同步部分，同时还包含并行运行的逻辑。这意味着并非所有操作都按顺 序依次发生，处理序列具有非确定性。由于布局通常由嵌套画面和面板组合构成，需注意：每个画面窗口和面板实例均 存在独立运行周期，为简化起见， 图 2-10 画面加载过程.中未可视化该周期。此外，可视化部分与脚本任务在并行逻辑 块中协同工作。

此步骤完成后，系统根据画面是否曾被打开过，从缓存或RDF文件中构建基础画面。随后在画面上定位面板。此步骤完 成后应用所选样式，并为所有属性（包括动态属性，如关联变量的属性）加载静态初始值。此时渲染过程启动并延续至 后续步骤。因此根据后续并行路径的执行时机，部分画面对象会先于其他画面对象显示。

HMI逻辑与脚本执行并行。因此，根据动态变量注册的A1模块与脚本变量注册的B1模块的执行速度，动态化处理 的A2模块或脚本执行的B2模块将率先启动。两者执行顺序不具确定性！根据项目和画面不同，变量动态化处理或 加载事件可能优先执行。

当两条逻辑均完成后，画面将被标记为"激活"状态当属性首次从静态值切换至动态变量时，属性变更事件将被触发。若随后通过并行路径触发的脚本修改关联变量，则画面加载过程中该属性变更事件可能被执行两次。下图中橙色高亮部分即为第二次变更事件调用 (图 2-11).

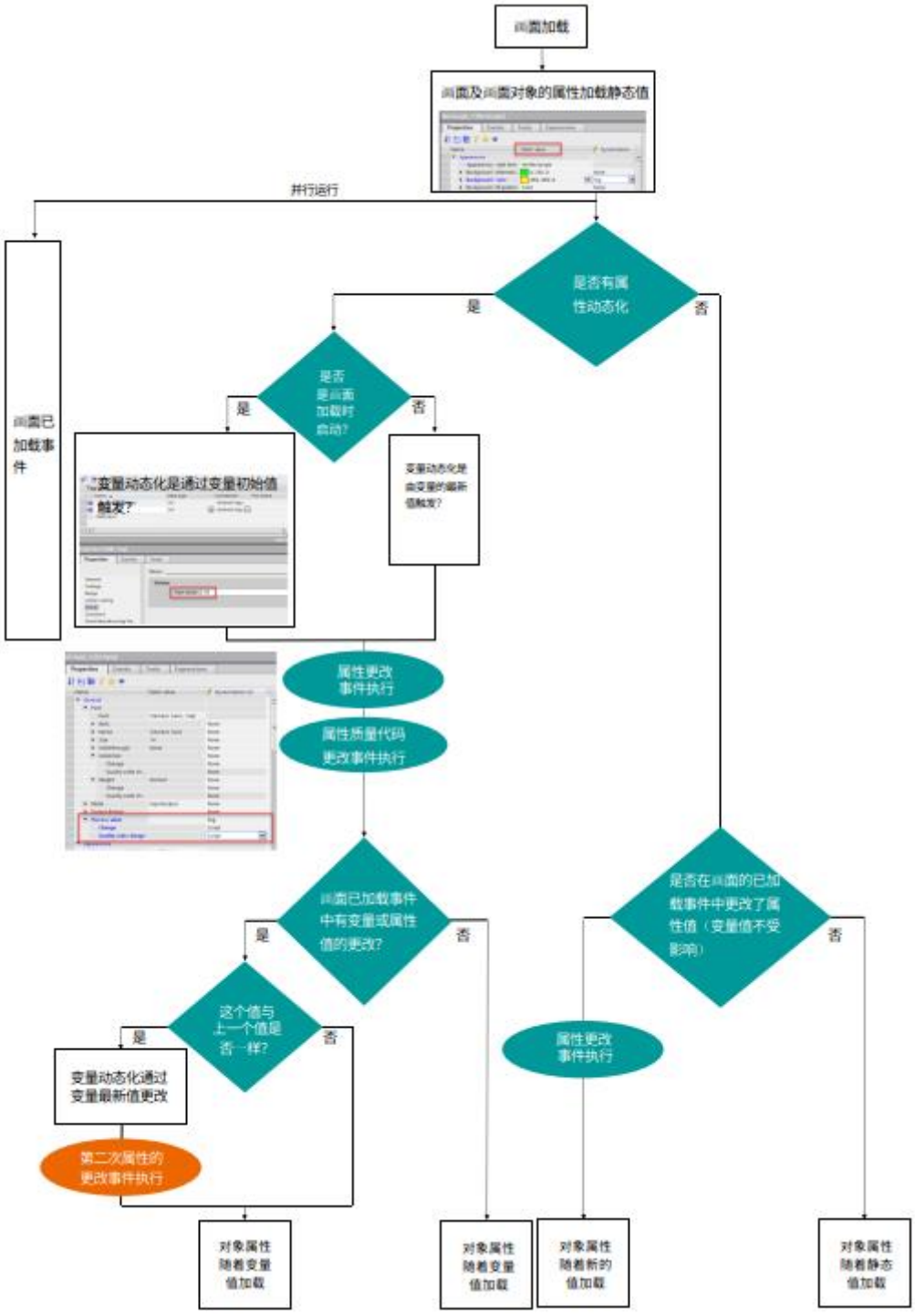


图 2-11 画面加载期间画面对象属性的已更改事件调用

下图显示了使用 PLC 变量进行动态化时的不同行为。重点是 PLC 采集周期时间和质量代码事件的第二次执行。有关采集周期的一般信息，请参考： 采集周期。

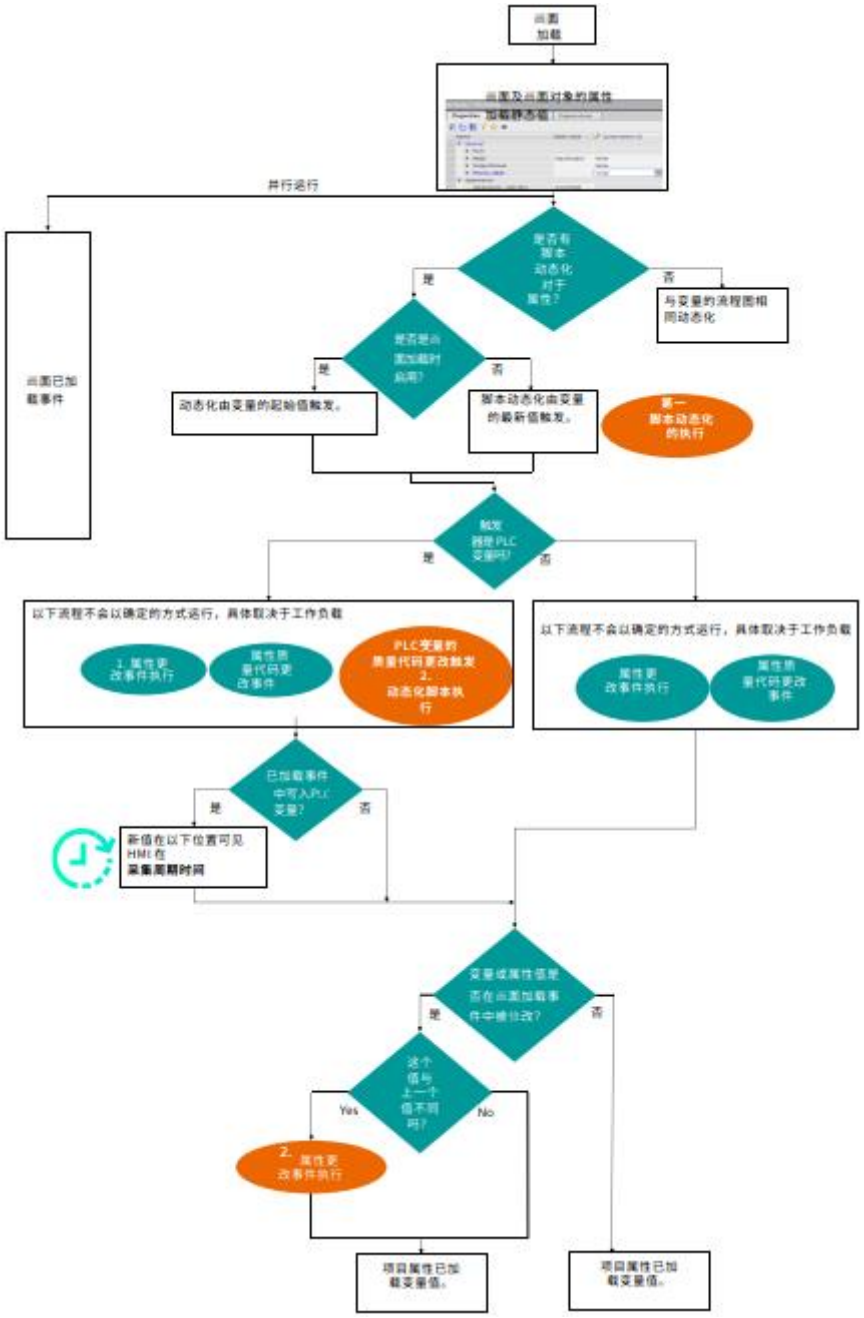


图 2-12 使用 PLC 标签进行动态化

NOTE

质量代码仅在画面加载时（且仅当已完成完整的下载时）进行评估，不再因画面切换而触发。
因此，由画面切换引发的不安全代码的第二状态将不再触发。 V20

另一个重要方面是执行上下文。上下文指的是执行特定脚本或任务的独立运行环境。不同的上下文在 Runtime 中并行运行，因此可以同时处理多个脚本和任务。

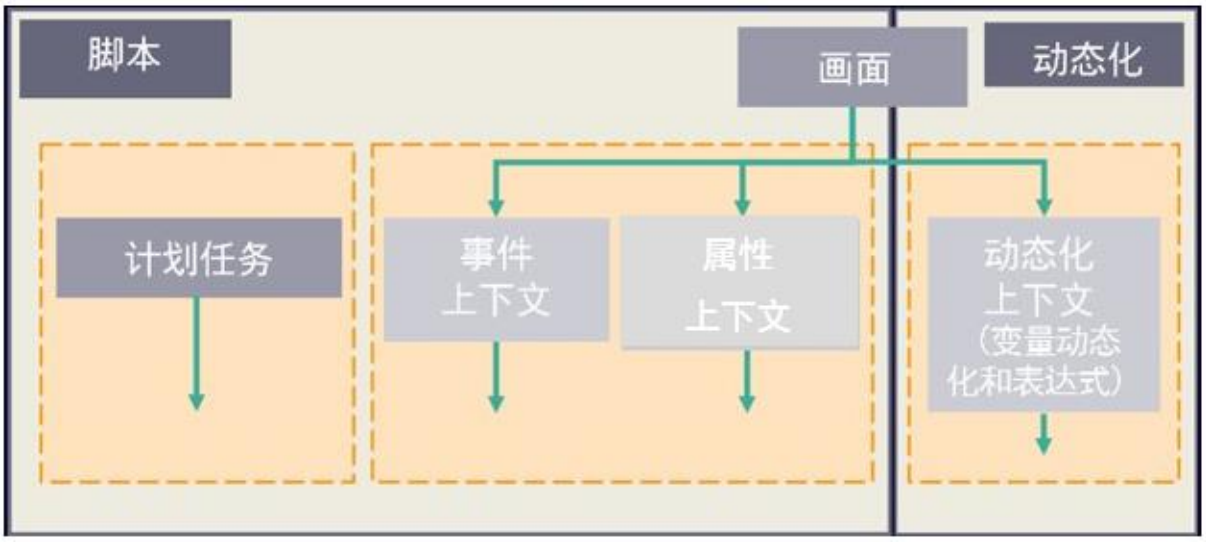


图2-11 脚本和动态化执行上下文

如上图所示，主要分为脚本和动态化两个部分。这已经说明了为什么变量动态化和表达式优于脚本动态化，因为它们与所有处理的脚本并行工作。
就脚本而言，主要分为计划任务上下文和画面上下文。每个画面都有自己的脚本上下文，用于脚本和系统功能。每个脚本上下文的命名空间（全局定义区域）都是唯一的。计划任务和画面上下文中的脚本可以循环触发、变量触发或报警触发。这两者对于加载过程都很重要，决定脚本是否在计划任务或画面上下文中实现以及使用哪种触发类型（见3.4.1章节）。

有了这些信息，就可以确定哪些配置与画面加载相关。

脚本	是否相关	更多信息
画面加载事件	✓	图 2-8 画面加载程序
画面全局定义区域中的脚本	✓	
计划任务	-	图 2-11 脚本和动态化执行上下文
脚本模块（参考）	✓	
“点击鼠标左键”事件中的脚本	-	
属性的更改事件脚本	✓	图 2-11 在画面加载过程中调用画面对象属性更改事件
动态化脚本	✓	

表 2-1 画面加载相关脚本概览

对象/实施	是否相关	更多信息
面板接口	✓	3.3 面板的使用
表达式	✓	
动态化	✓	
使用的 HMI 变量数量	✓	见 Unified 手册中的系统限制
画面元素数量（项目、面板实例、画面窗口 .....）	✓	见 Unified 手册中的系统限制
画面元素类型	✓	3.1.1 画面对象选择

表 2-2 画面加载相关对象概览

正如所看到的，影响 Runtime 设备画面更改过程的因素有很多。为了向您提供所有单个方面的详细信息，我们将更加关注以下几个方面。同时，您还将获得如何以最有效的方式实现这些功能的说明和最佳实践。如果需要对项目进行分析，第 4 章 将提供指导。但首先，本文档将详细介绍画面工程的最佳实践。

图 2-12 最佳实践主题概览

3. 画面工程最佳实践

本章节将介绍许多关于 HMI 项目组态的建议。因此，我们将首先向您介绍可以在哪些方面施加影响，以及有哪些可以提升的可能性。然后，我们将通过使用案例为您提供详细说明。我们将按照画面对象、动态化、面板、脚本和其他方面的使用来划分章节。

注意 为帮助理解这些最佳实践，第 2.4 章介绍了常规屏幕加载程序。

3.1. 画面对象的使用

下图显示了PC RT Unified 的工具箱中可用项目（Unified 面板的内容可能有所不同）。一般来说，有五个部分：基本对象、元素、控件、我的控件和动态部件。

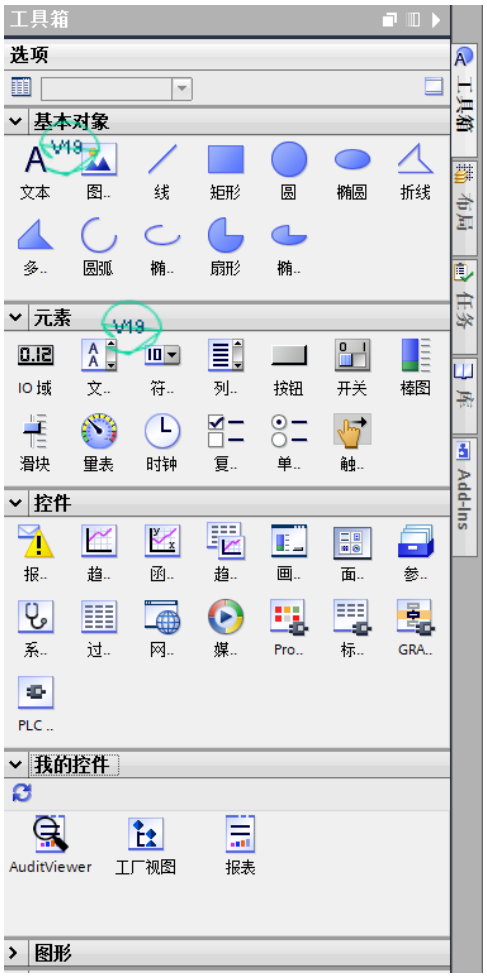


图3-1 PC-RT 的 Unified 工具箱 V19

一般来说，本节有关画面对象的所有建议都遵循这一规则。

画面对象的使用

尽量减少画面上的对象数量，选择占用空间最小的对象

在我们继续之前，有必要提及每个对象都有一个占用空间。占用空间描述了渲染的工作量。占用空间小的对象属性数量少，画面项目（如矩形）的复杂度低。占用空间较大的对象通常包含大量属性，如控件。除了固定属性外，通过动态化进行定制也会增加画面对象的渲染工作量，从而影响整个画面的加载过程。

3.1.1. 画面对象选择

首先，在画面上放置新对象时，应选择占用空间最小、仅包含所需功能的对象。一般来说，从基本对象到控件，复杂程度会随着工具箱中的顺序而增加。下图按顺序显示了所有画面对象所需渲染工作量。请注意，本图中显示的渲染工作量并不与时间等效，只是显示关系。实际渲染工作量还取决于配置（动态化和连接的数据）。

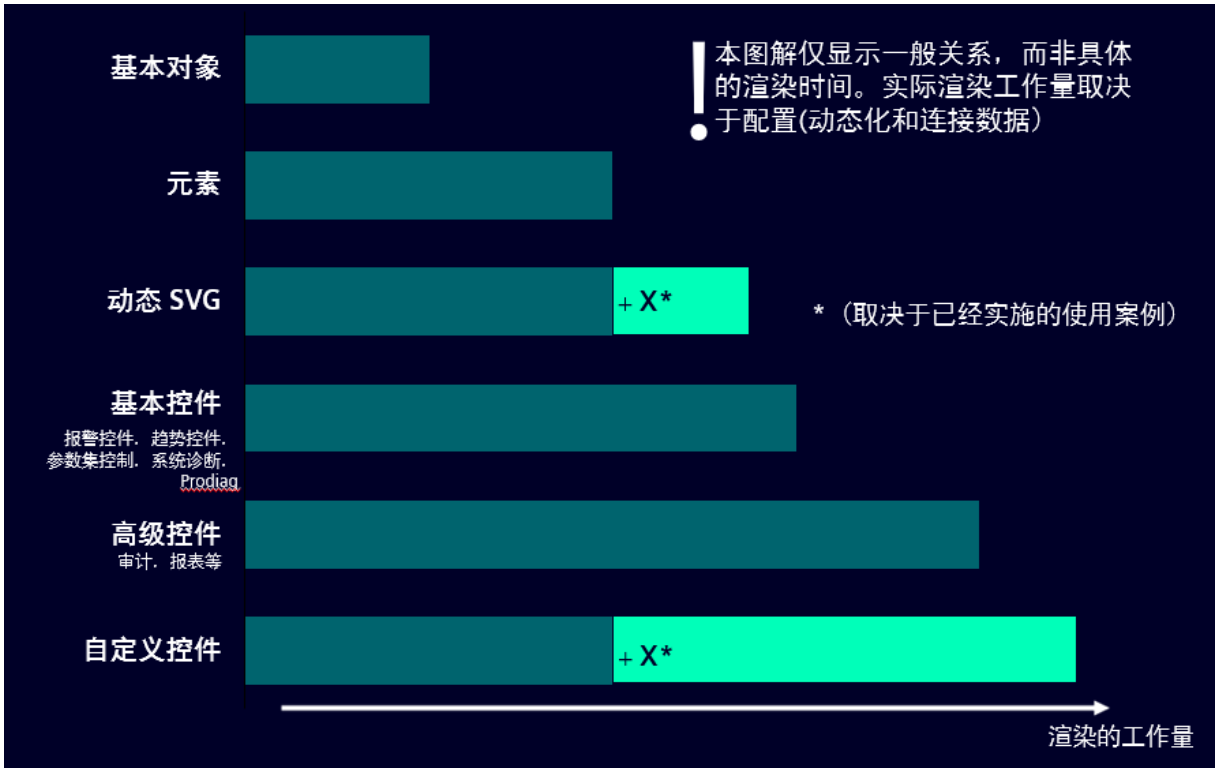


图3-2 画面项目渲染工作量的相对比较

组本身也存在差异，最重要的是要了解每个画面对象都有各自的渲染工作量（占用空间）。在我们继续之前，请确保您了解工具箱元素，甚至是 V19 中的更改。文本和文本框的更改如下图所示。

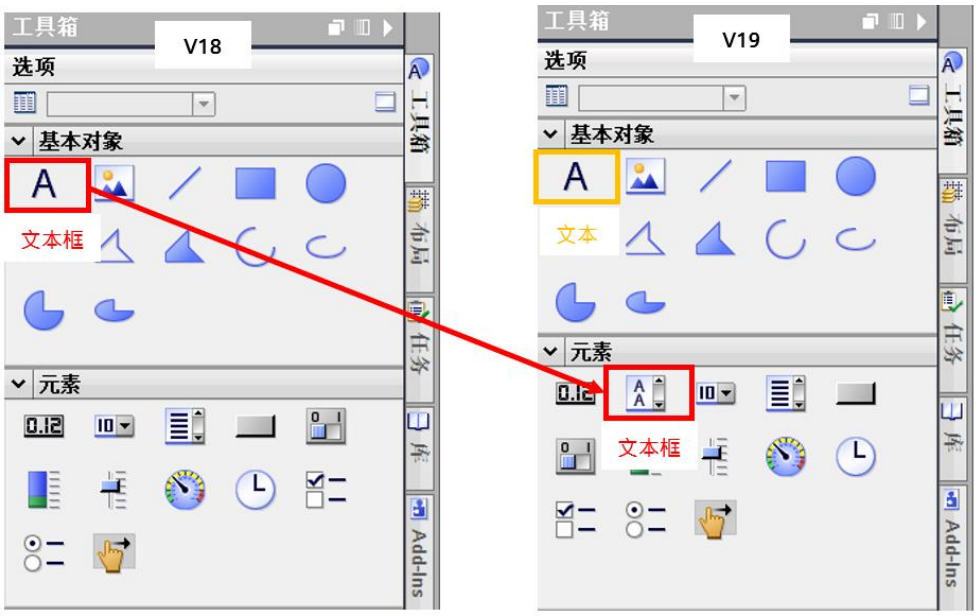


图3-3 V18 的文本框 与V19 的文本框

注意 V18 V18 中基本对象中的文本框已移至 V19 的元素部分。在 V19 工具箱的基本对象中，同样外观的项目现在"仅 "是一个文本，相当于一个简单的标签。因此，如果工程系统是 V18 版本，文本框的渲染工作也会比基本对象高，与元素部分的项目相当。

有时，用不同的对象显示一个功能的可行性不止一种。即使两个对象在画面上的配置看起来一样，它们对整个画面的加载时间也会有不同的影响。下面将举例说明。

即使下面的示例没有涵盖完整的实际用例，在配置新画面时也要牢记这些要点。

3.1.1.1. 使用案例：彩色的方框

说明

如下图所示，本用例是关于彩色方框的配置。可能的解决方案是矩形或不带文本的文本框，因为它们在画面上看起来是一样的。

解决方案

与使用文本框相比，矩形元素的渲染工作量更小，是更好的选择。

图3-4 矩形与文本框

画面对象选择

始终使用占用空间最小且具有必要功能的画面对象

3.1.1.2. 使用案例：不执行按下和释放事件的按钮

说明

本用例涉及按钮的配置，该按钮只使用"单击鼠标左键"事件，而不执行"按下"或"释放"事件。
可能的解决方案包括文本框、文本 (V19) 或按钮。

解决方案

就渲染工作而言，文本占用的空间最小，其次是文本框和按钮。因此，如果您需要背景颜色，请使用文本框，否则，如果文本能满足您的要求，请使用文本。而且，这些对象都有 "单击鼠标左键" 事件。只有在需要"按下"和"释放"事件时才使用按钮。

图3-5：文本作为按钮

图3-6: 文本框作为按钮

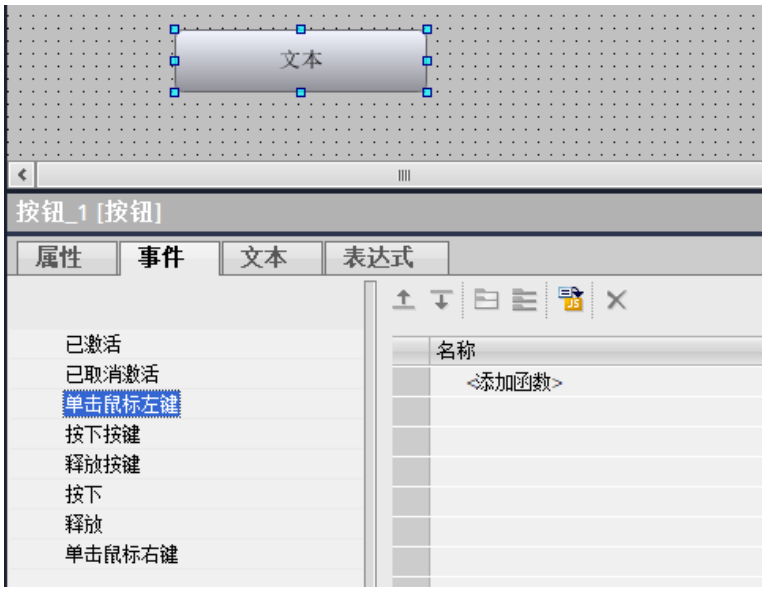


图3-7：按钮作为按钮

3.1.1.3. 使用案例：简单文本 V19

说明

对于画面上的文本元素，可以使用文本框或文本。

解决方案

只要有可能，特别是对于简单的变量，请使用文本而不是文本框。下表显示了文本与文本框的不同功能集。与文本框相比，文本的功能范围更小，因此在Runtime 设备上占用空间和渲染的工作量也更小。

属性	文本	文本框
颜色	前景	前景、背景、边框、交替颜色
边框	X	颜色、宽度
格式 - 间距	X	✔
格式 - 文本截断	X	✔
格式 - 文本换行	X	✔
杂项- 连接状态	X	✔
杂项- 只读	X	✔

参数字段	✔	✔
可编辑/输入文本（RT 中）	X	✔
复制并粘贴（在 RT 中）	X	✔

表3-1 画面项目文本和文本框的比较

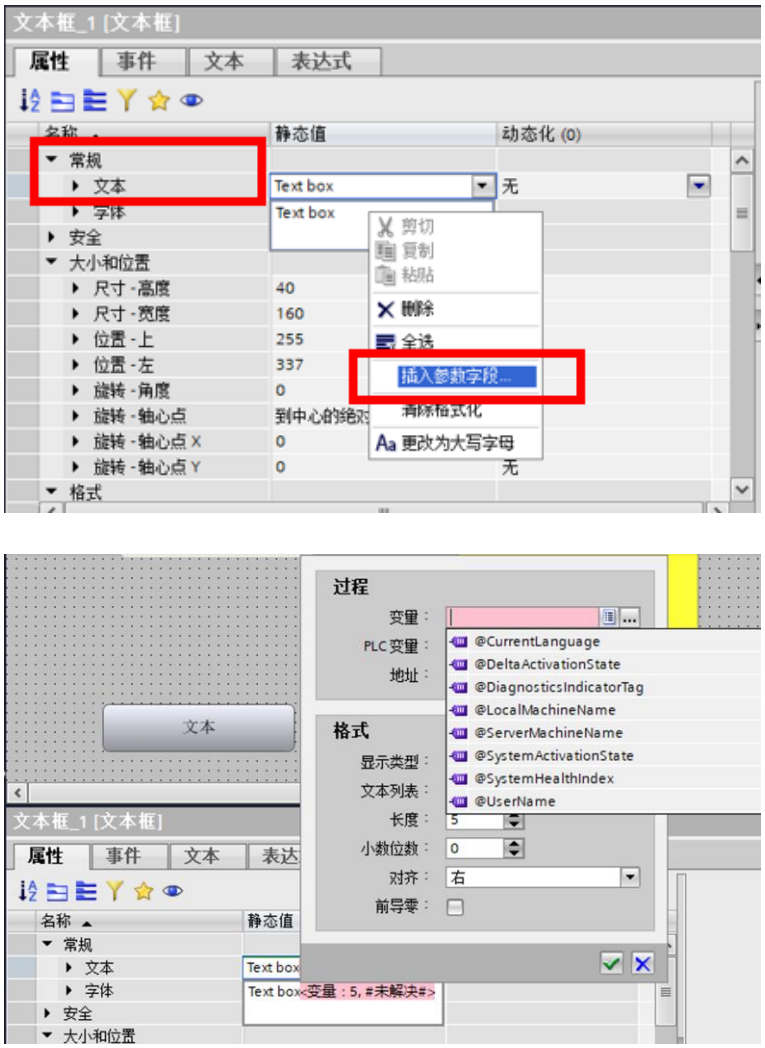


图 3-8 文本属性的参数字段

3.1.1.4. 使用案例：彩色画面背景

说明

在某些情况下，需要调整画面的默认背景颜色。

解决方案

与其在画面上添加一个额外的矩形并使用其背景色作为画面背景，不如直接更改画面背景的颜色。这也适用于面板。

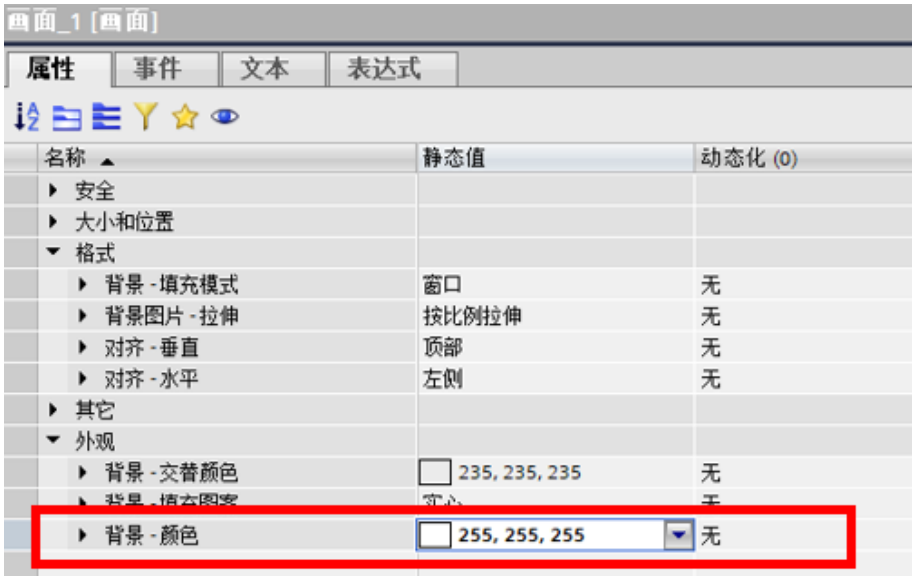


图3-9 画面元素的背景色配置

3.1.1.5. 使用案例：自定义样式 V19

说明

通常情况下，一个企业布局也会有一个企业调色板和设计。通过对象属性，画面对象可以适应这种布局和设计。此外，按钮等对象的整体外观也可以成为企业设计的一部分，甚至可以是滑块等更复杂的对象可视化。

解决方案

最好使用 Unified 的画面对象来实现其预期功能。应避免通过多个画面对象的来实现用户自定义的设计，因为这样会导致画面对象的数量增加，通常还会增加这些辅助对象的未用功能。此外，使用面板作为企业设计对象的标准化方法也会造成不必要的过载。使用 SIMATIC WinCC Unified Corporate Designer 可以创建这些设计对象和颜色，作为您的项目风格的一部分。

在 Corporate Designer 中，可以在现有样式的基础上创建新样式。这样，既可以更改现有对象的样式，也可以创建全新的对象。这些样式还可以与调色板和字体相连接。

样式设计完成后，需要导出样式，并将生成的文件复制到自己新创建的项目目录（Userfiles > Styles）。完成这些步骤后，就可以在runtime设置中选择并使用该样式。

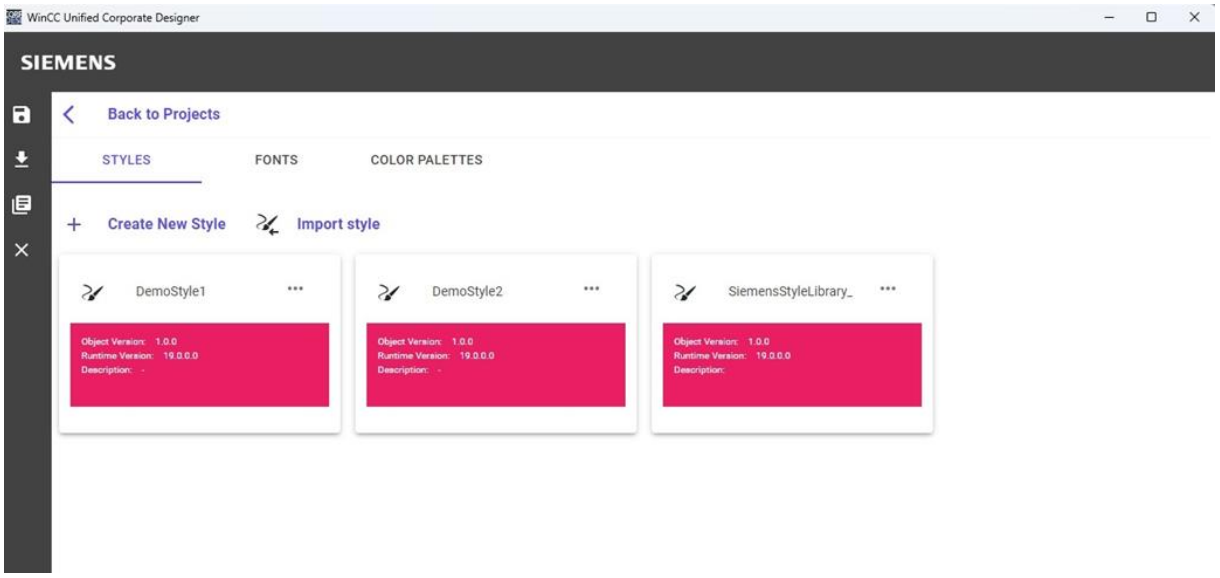


图 3-10 Unified Corporate Designer 项目视图

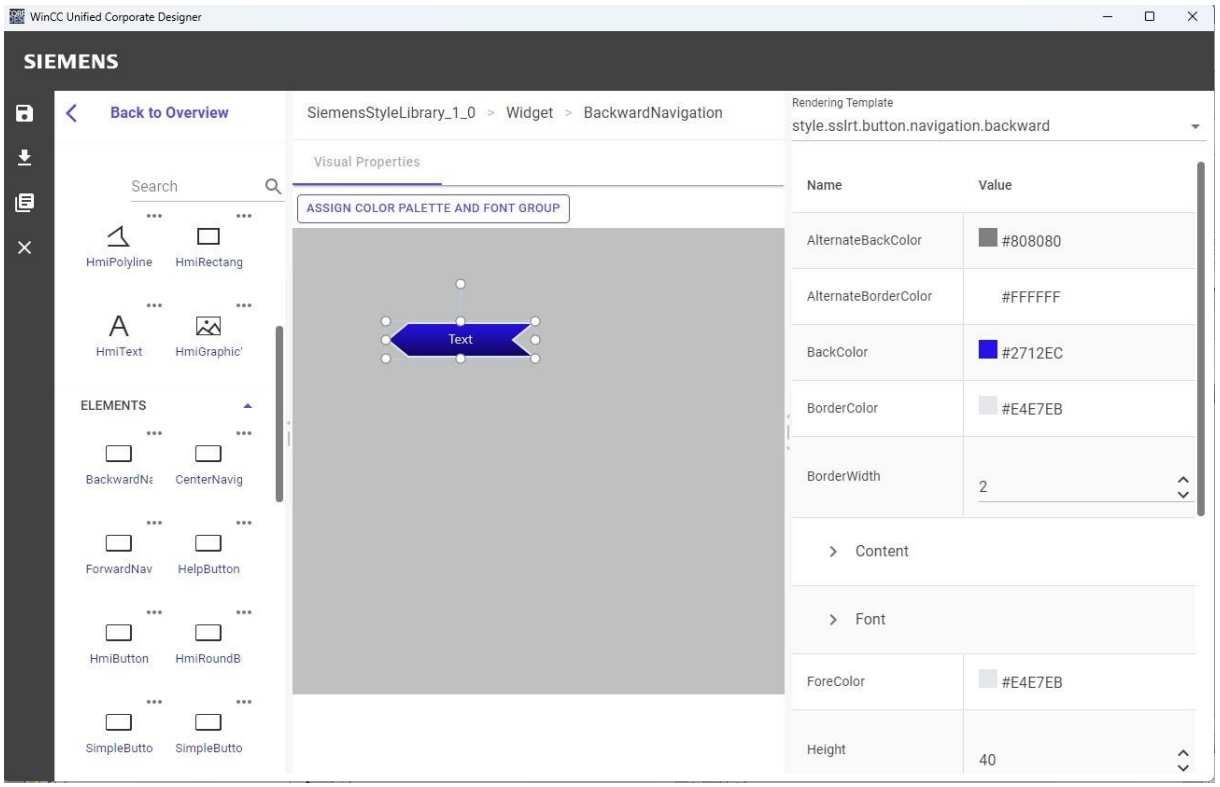


图 3-11 Unified Corporate Designer 编辑视图

在设备 Runtime 设置中已经选择了的样式，仍然可以在 Runtime 中按照需更改。此外，还有新的自定义样式。

图3-12 Unified Runtime 的样式选择

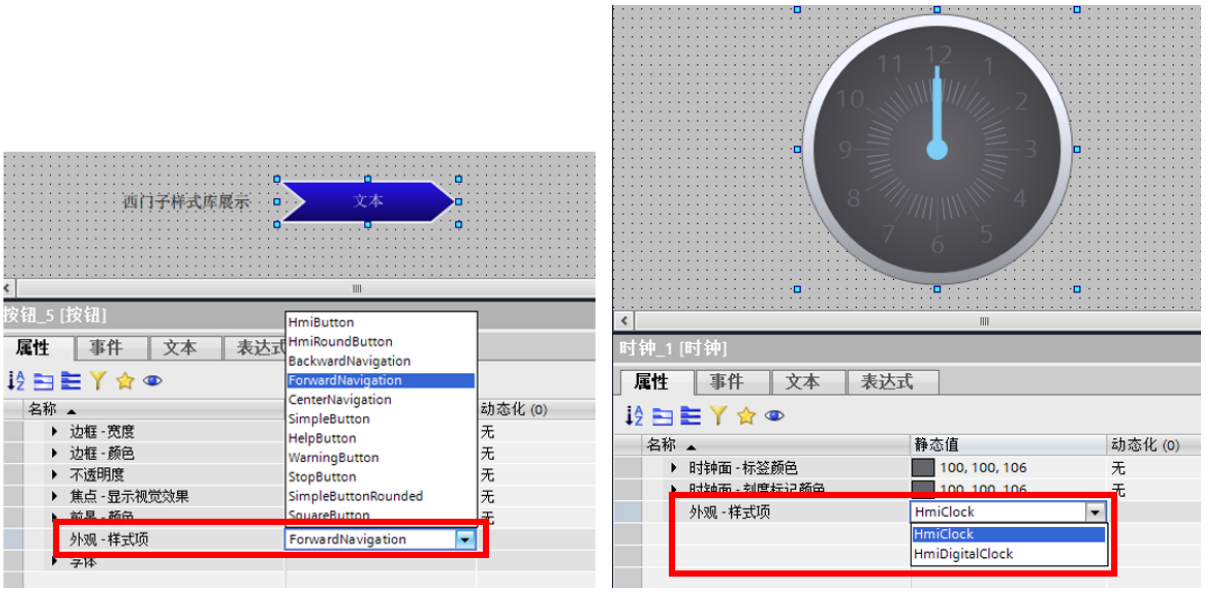


图3-13 在工程组态中更改系统和自定义画面对象的样式

自定义样式

为单一对象配置定义自定义样式，而不是使用面板

3.1.2. 整理画面/遵守系统限制

正如看到的，有些画面元素需要更多的渲染工作，而有些元素则需要较少的渲染工作。无论如何，画面上的每个元素都会影响加载时间，而且还必须遵守已被定义的系统限制。为了实现最短的画面加载时间，尤其是在项目即将用于生产之前，应再次删除所有仅在实施阶段因调试或其他辅助原因而生成的对象。

- 删除后台或不可见的未使用对象
- 删除超出画面范围的未使用对象。布局部分有一个任务卡，也可以显示超出范围的对象。

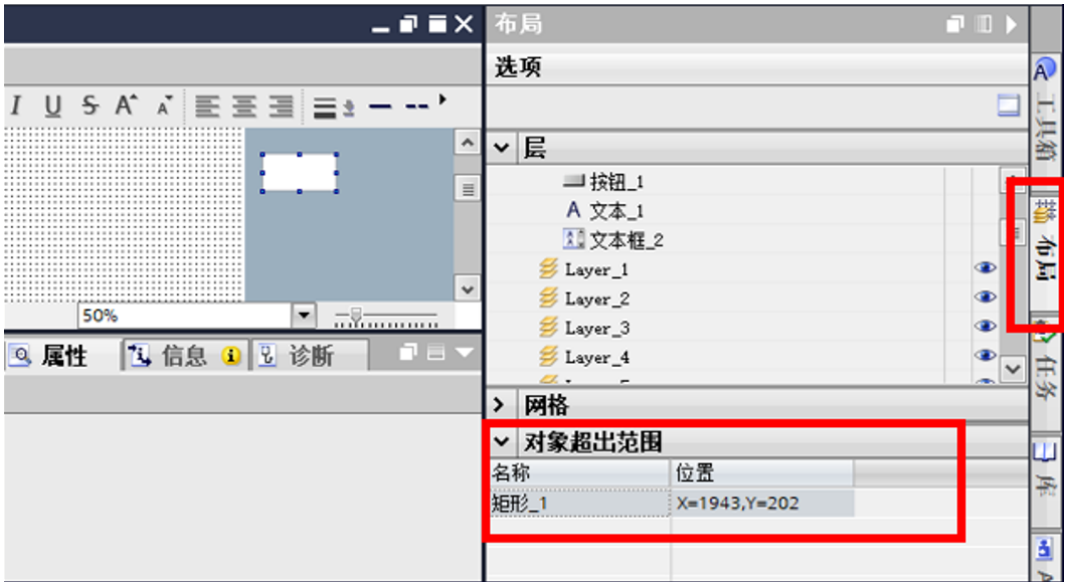


图 3-14 任务卡找到超出范围的对象

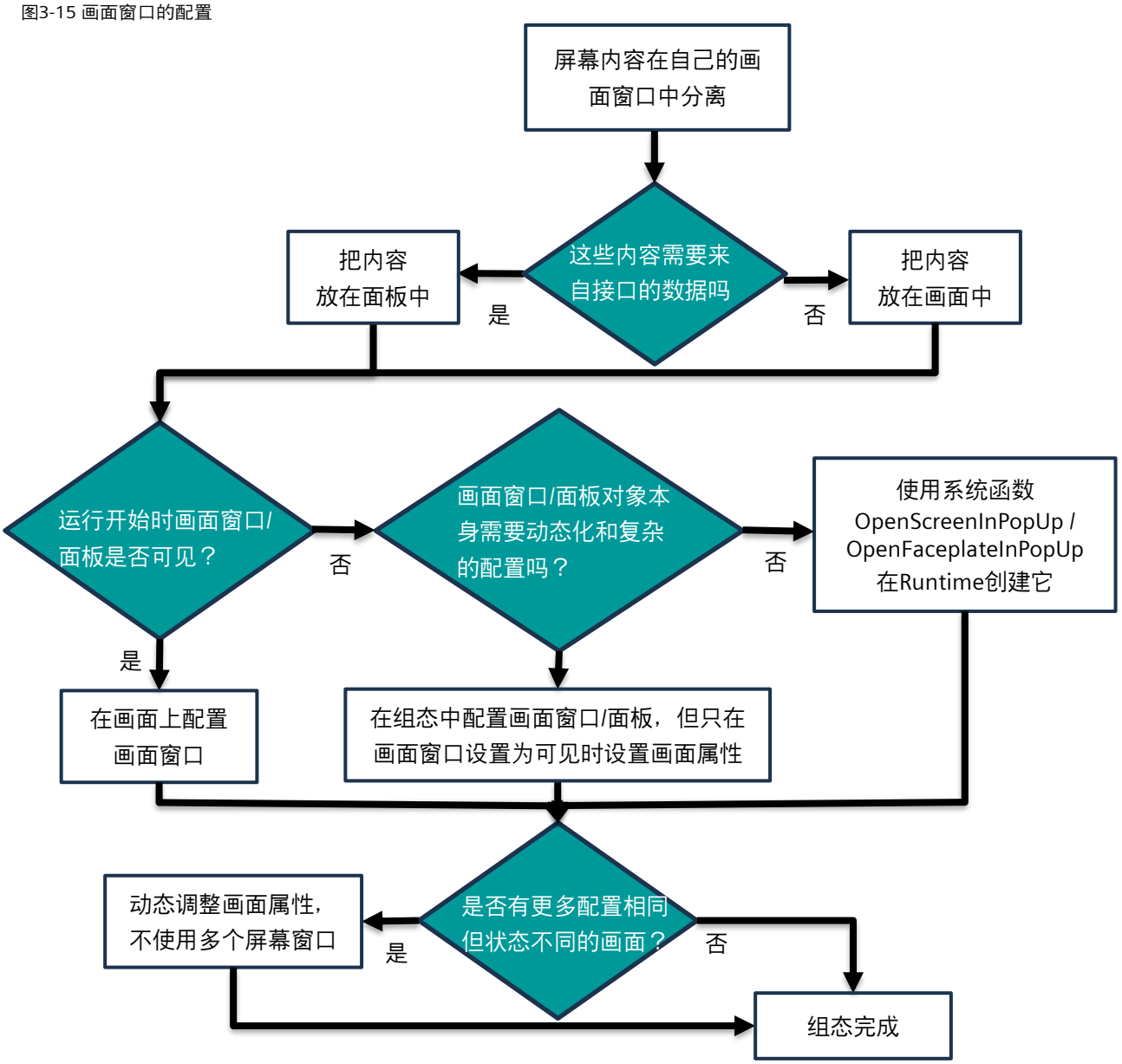
系统限制

删除未使用和不可见的对象，以降低加载工作量

3.1.3. 画面窗口的使用

画面窗口通常用于创建画面布局，并将单独内容与可用于更多视图画面且无需在每次画面变化时加载的内容（如标题栏）分开。此外，弹出画面也会使用画面窗口。为明确何时将画面窗口用作弹出窗口，何时在 Runtime 对其进行配置，请参考下面的流程图。本建议仅从好的工程设计角度出发。请注意，一般情况下，画面窗口在缩放、滚动、位置、画面层和实时方面与系统功能 OpenScreenInPopUp 生成的弹出画面具有不同的行为。因此，特定用例可能需要明确采用哪一种实现方式。

图3-15 画面窗口的配置



3.1.4. 画面对象可见性

如果画面对象的可见性是动态的，则应将静态值设为 false，特别是当可以认为所使用的动态化（例如变量动态化）在画面加载期间导致不可见对象时。

图3-16 如果画面对象在加载时不可见，可见性的静态值应设为 false

3.1.4.1. 使用案例：根据相同条件动态调整多个画面对象的可见性

说明

在Runtime，需要通过同一条件改变多个画面对象的可见性。

解决方案1

如果多个对象的可见性根据同一条件发生变化，则定义一个变量来动态显示所有对象的可见性。

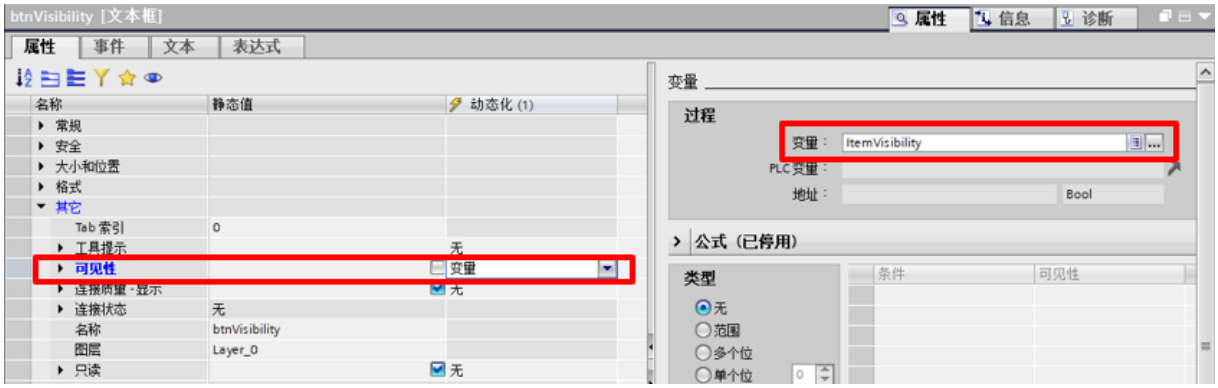


图3-17 画面对象在Runtime 可见性

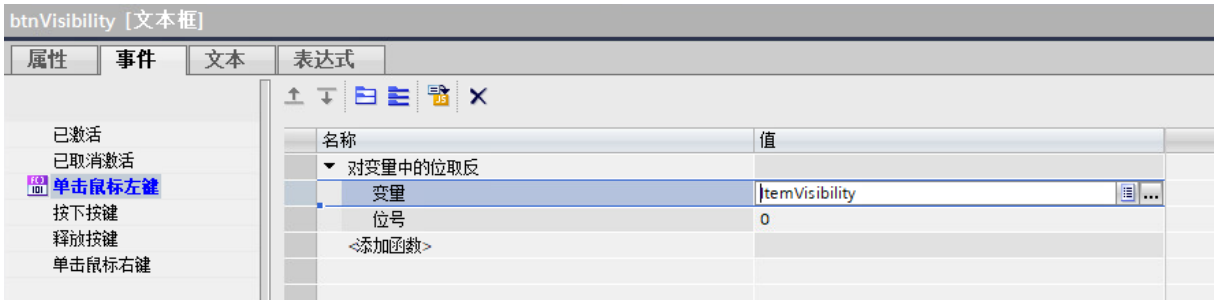


图3-18 切换对象可见性

解决方案2

如果只是根据相同的条件改变可见性，则将所有对象移到一个层中，并动态调整该层的Runtime 可见性。也可以通过脚本在Runtime 更改层的可见性。

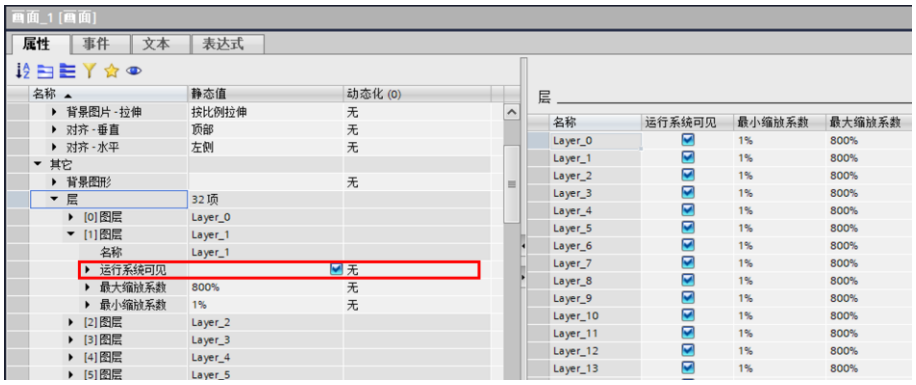


图3-19 画面层级在运行系统上的可见性

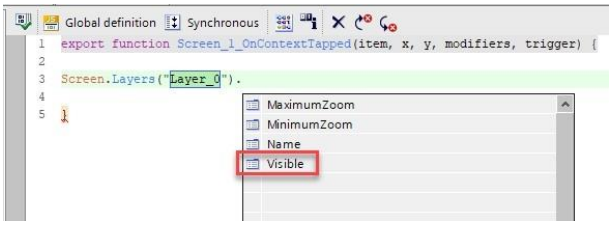


图3-20 访问运行系统中画面层级的可见性

解决方案3

如果有更多的属性具有相同的行为或条件（如背景颜色、权限等），则将对象分组并一次性为整个组配置属性。

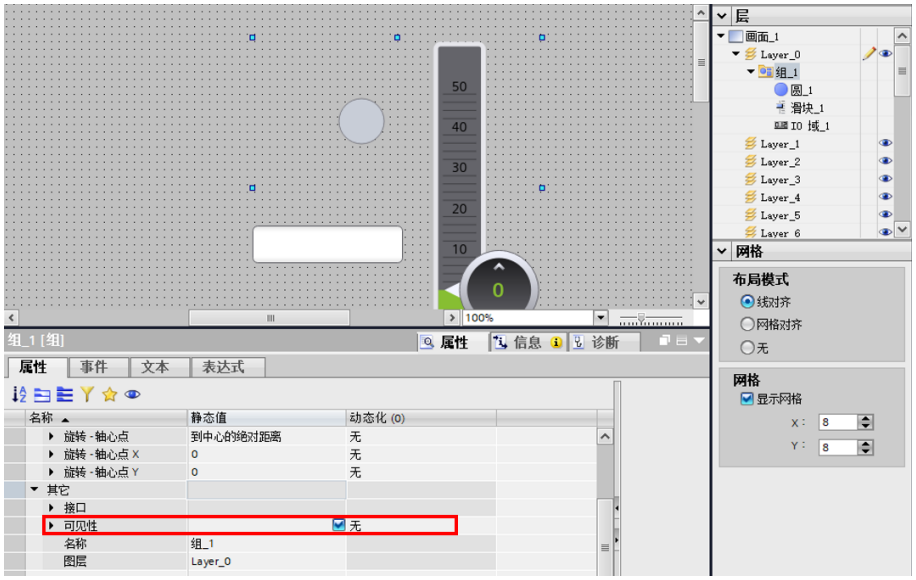


图3-21 一组画面对象的可见性动态化

画面项目的可见性

如果多个项目的可见性被同一条件动态化，可将它们放在一个组或层中并动态化其可见性。

3.1.5. Unified 控件

Unified 控件可以更容易地实现报警、报表和趋势控件等常见用例。不过，在使用过程中也有可优化的地方。

3.1.5.1. 使用案例： Unified 控件在 Runtime 上的不同设置

说明

在工程组态中，可以对 Unified 控件进行大量配置。有时需要在 Runtime 上对控件进行不同的设置。由于以前在 WinCC Basic、Comfort 和 Advanced 项目中动态化控件的可能性非常有限，通常的做法是在这些项目的画面中放置几个静态参数化控件，并在 Runtime 上交替显示。

WinCC Unified 提供了非常广泛的可配置和可动态化的控件，因此只需实施和控制一个控件，就能满足在Runtime 提供各种设置的要求。

解决方案

在 Runtime 上更改属性，不配置多个控件，甚至不更改画面来实现不同的设置。因此，请使用系统函数 "SetPropertyValue"或对象模型"Screen.Items("Itemname")"。属性名称可以很容易地从属性选项卡复制。

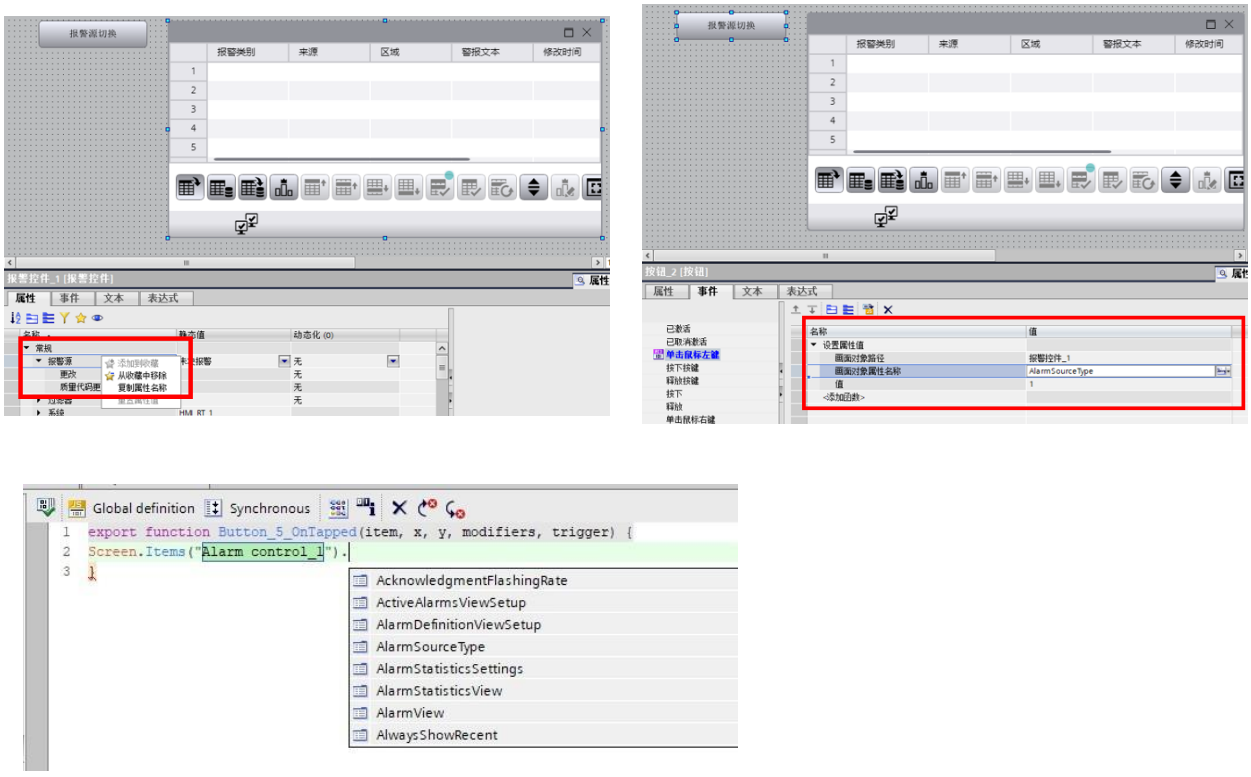


图3-22 访问 Unified 控件属性

所有可访问的属性可直接在 WinCC Unified 对象模型的 TIA 帮助或 SIOS 上的 SIMATIC HMI WinCC Unified EngineeringV20 系统手册中找到。

Unified 控件

当需要不同的设置时，可在 Runtime 重新配置控件，而无需使用多个控件。

3.1.5.2. 使用案例：报警控件

说明

通常，为了访问报警，提供了报警视图。该控件在工程中以及在Runtime 都有许多功能需要配置。此外，还有很多系统函数。可以通过脚本访问相同的功能，而无需控件。

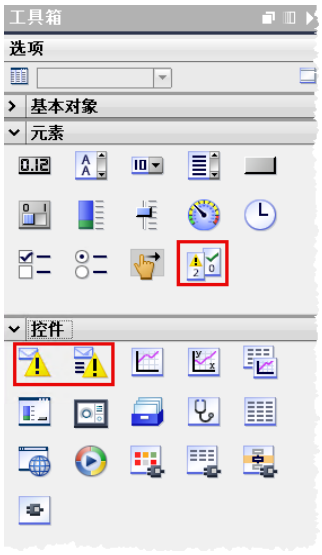


图 3-23 Unified报警控件

一般情况下，我们建议使用画面窗口来设计画面布局（见 3.1.1 章节和图2-2），因为有的画面会更改内容，有的画面则保持不变。

解决方案

报警控件占用较大的显示区域（见3.1.1 章节和图 3-2），因此建议将其放在不经常重载的画面窗口中，例如标题栏。

使用报警视图

- 如果在HMI画面运行操作期间画面窗口不经常重新加载
- 如果画面窗口足够大，可以容纳整个报警视图
- 如果需要报警视的完整功能
- 如果需要具有完整功能的报警视图，请根据需要将其设置为弹出窗口显示

以下示例演示了同一个报警控件显示两个不同设备画面的用法。当切换画面到另一个设备的画面时，报 警控件会按来源进行过滤，仅显示相关的报警。

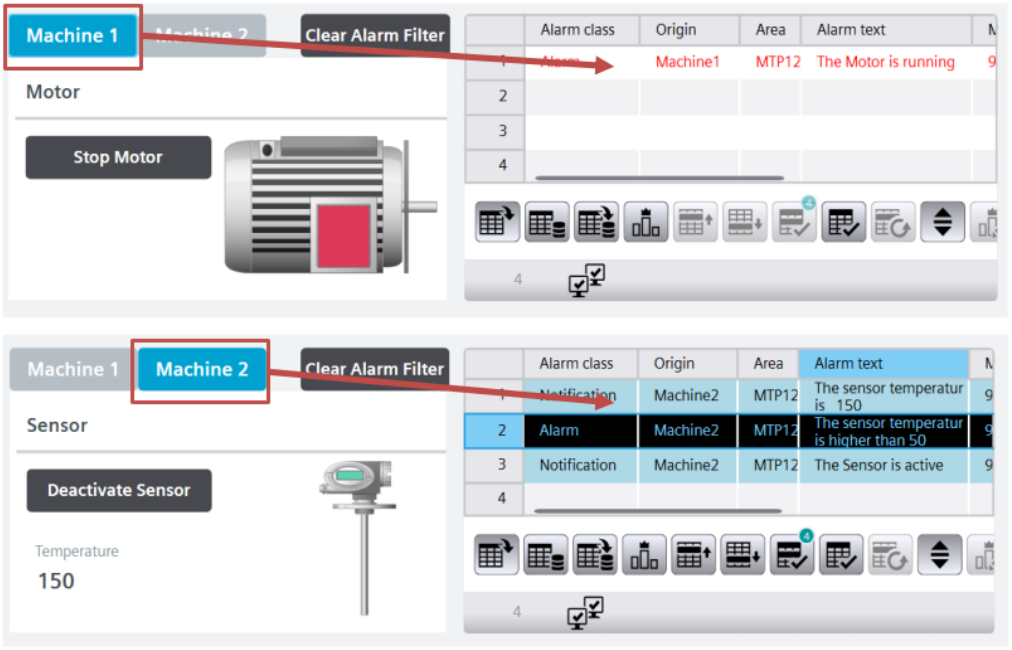


图3-23 于两个不同画面的报警控件

解决方案

使用报警行

- 如果您需要在频繁切换的系统中始终显示最新的报警信息
- 如果您需要在像标题这样的小画面窗口中使用报警行

在以下示例中，报警行在画面中使用了三次，并按报警来源进行过滤，以及显示 与指定机器相关的信息。

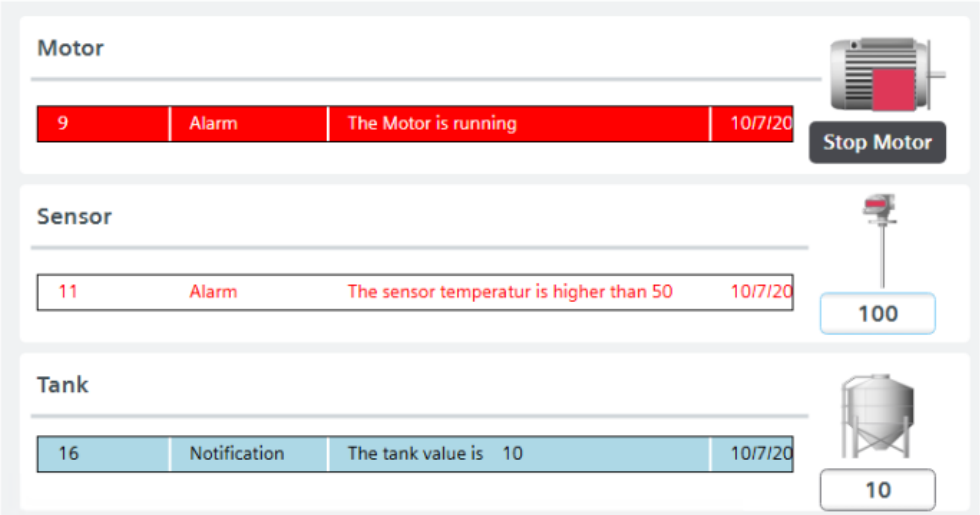


图3-24报警行应用

3.1.6. 图形和 SVG

除了所有文本信息外， HMI 通常还包含图形。这些图形可以是小图标、大图形或动画图形/动态SVG。根据不同的使用情况，也有关于如何使用图形的建议。

图形格式

- **JPG**（联合摄影专家组）格式，建议将 JPG 用于 Unified 精智面板，因为这种格式可以减小大文件的大小。虽然压缩会导致质量下降，但文件解码速度也会加快。当需要高分辨率的图形时，也可以使用高质量的JPG。JPG 并不意味着自动降低质量。
- **PNG**（便携式网络图形）是一种无损数据压缩的光栅图形格式。在需要透明度或需要精确像素时，建议使用PNG。
- **SVG**（可缩放矢量图形）用于在网站上显示二维图形、图表和插图。作为一种矢量格式， SVG 图形可缩放至不同大小而不影响其分辨率。因此，在需要高质量缩放或使用动态 SVG 时，建议使用这种格式。在其他情况下，最好将所用尺寸的文件转换为PNG，如果不需要透明度，则转换为JPG，其中应选择用于多次使用的最大尺寸。

注意

这些格式可分为两类：

- **光栅图形格式**，依赖于预先渲染的位图，因此对系统资源的要求较低，建议用作 **Runtime 格式**（JPG、PNG）
- **描述性图形格式**，使用代码指定图形的显示方式。它只在 Runtime 渲染，因此大小容易调整（如缩放时），非常适合作为主要的**库格式**（SVG）。

图形

无论采用哪种图形格式，文件大小都不应大于该人机界面所需的最大尺寸/分辨率

图形格式

建议选择图形格式的顺序是

1. JPG
2. PNG
3. SVG

如果对图形有进一步的要求，SVG 仍然是最合适的选择。但请谨慎选择！

3.1.6.1. 使用案例：可视化模式和组合对象

说明

有些物体或图案可以由多个画面对象组合而成。此外，这些组合对象还可以在画面上重复使用多次。

解决方案

不要使用单个画面对象来创建图案，而是将它们组合成一个图形。这样可以减少画面上的对象数量。即使是矩形等基本对象，组合成一整个对象也能减少渲染工作量。当需要动态化时，创建一个自定义的动态 SVG（见下一个用例）。这样也可以减少画面上对象容器的数量，并有助于遵守系统限制。

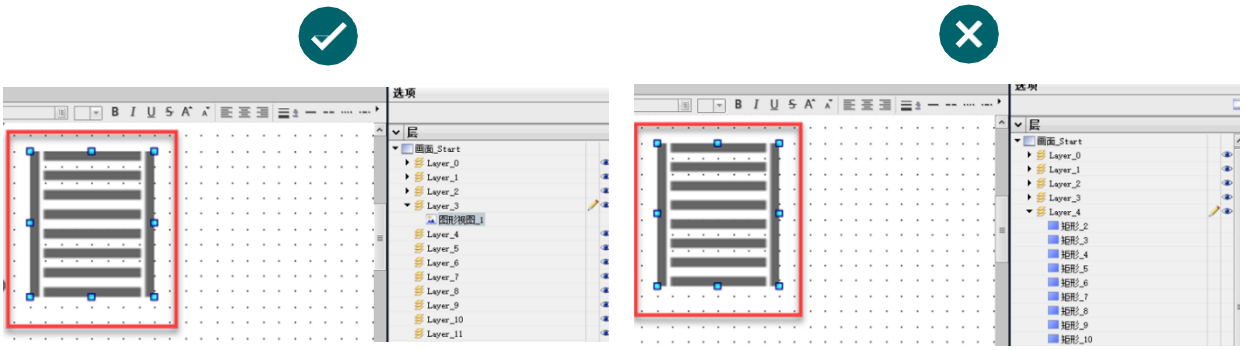


图3-35 SVG 样式示例

合并对象

通过创建SVG 和组合对象的动态 SVG，减少屏幕上的元素数量

此外，请使用原始分辨率的图形，并只使用必要的大图。

3.1.6.2. 使用案例：动态合成对象 → 动态 SVG

说明

一些动态可视化图形既可以通过多个动态画面对象的组合来创建，也可以直接通过动态 SVG 来创建。

解决方案

如前一个用例所述，为尽量减少每个画面的画面对象数量，应将组成的对象合并到一个对象容器中。工业图形库（IndustryGraphicLibrary）中有适用于各种标准组件的动态 SVG。在创建具有多个画面对象的组件之前，请检查图形库中是否已有解决方案。通过接口可以在 Runtime 动态配置和更改某些属性。有关动态部件使用的更多信息，请参阅动态小部件（Unified）。

图3-36 行业图库

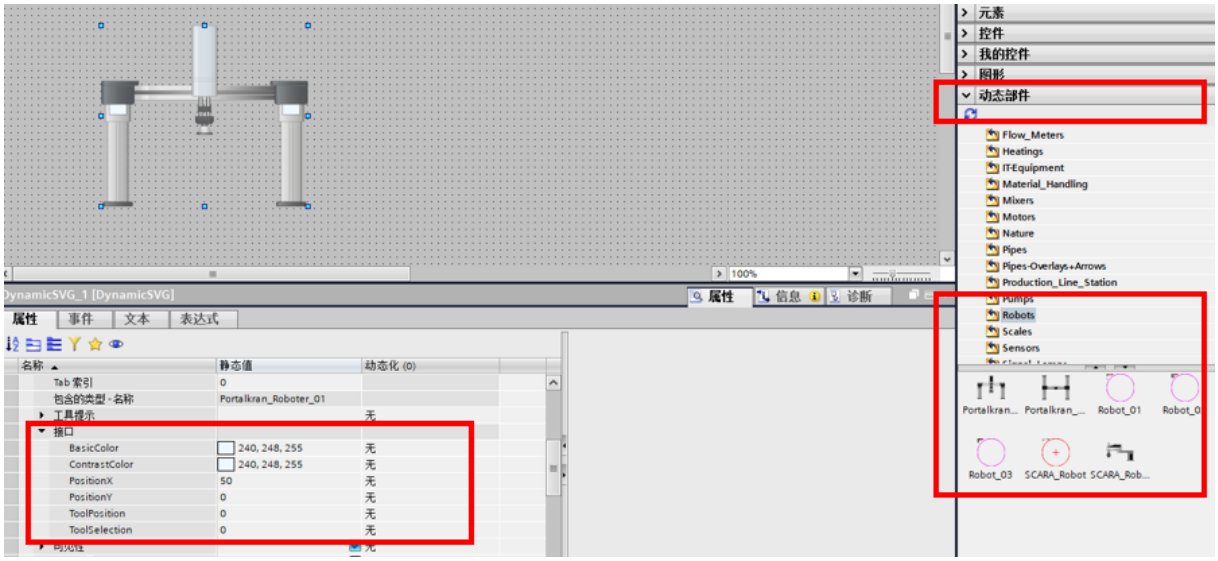


图3-37 西门子图形库中的机器人动态SVG

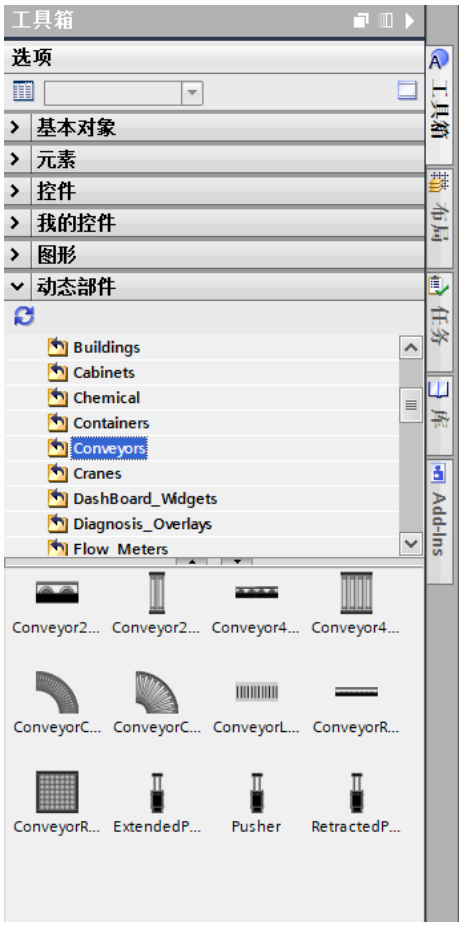


图3-38 动态部件传送带示例

3.2. 动态化的使用

本文件中提到了不同类型的动态化。

3.2.1. 简单变量动态化

变量动态化如下图所示。它可以有多种配置类型。可以使用“变量”直接将变量值转换为属性值。使用“范围”类型，可以为多个变量值范围指定一个条件，并将其链接到一个属性值。通过"多个位"类型，可以根据动态变量的不同位的值改变属性。最后，"单个位"类型仅限于动态化变量的一个位，用于指定属性的条件。每次变量值发生变化时，属性值也会随之调整。

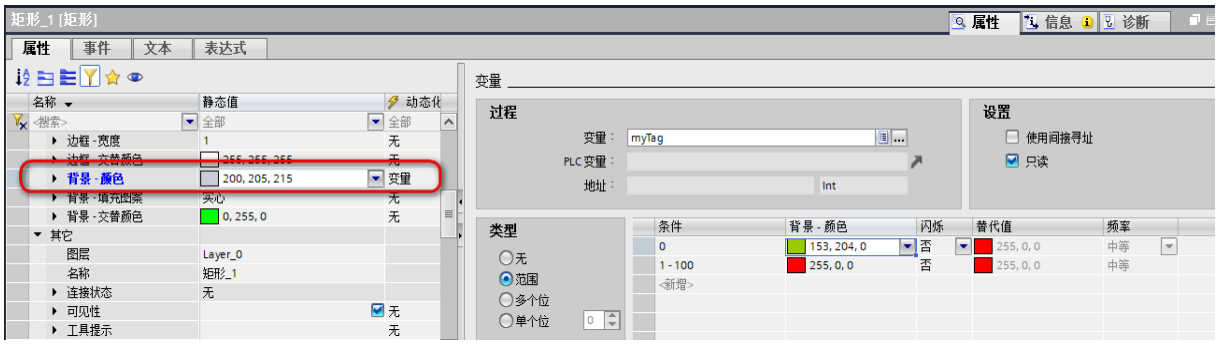


图3-39 变量动态化

在这种情况下，必须提及的是还有一个 "更改"事件。如果属性在 Runtime 上发生变化，例如动态变量的变量值发生变化，则可通过该事件执行附加脚本。在画面加载过程中触发该事件时发生的情况如图2-11所示。

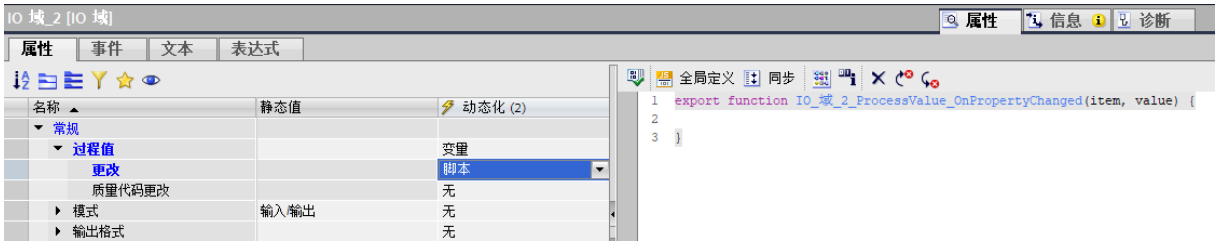


图3-40 变量动态化-更改事件

3.2.2. 脚本动态化

第二个动态化是脚本动态化。该脚本的返回值定义了脚本执行后的属性值。要执行脚本，需要指定一个触发器。有关触发器的信息请参见第 3.4.1 节。

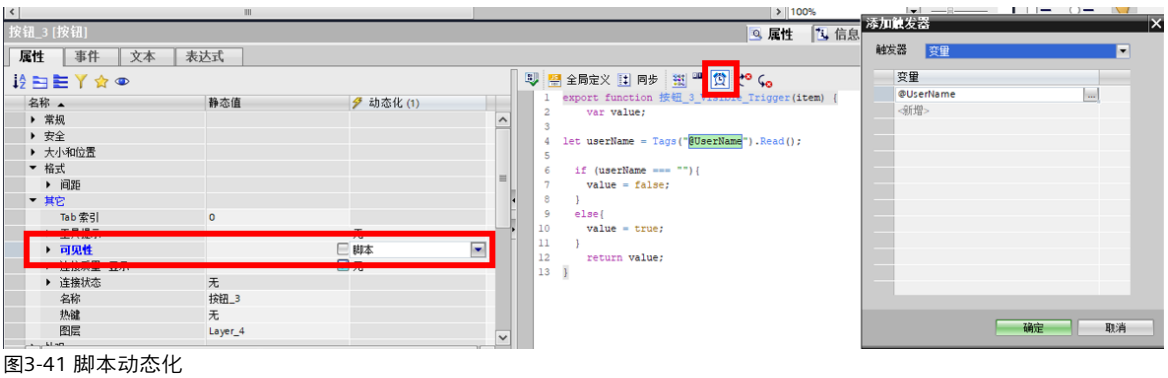


图3-41 脚本动态化

3.2.3. 表达式

表达式是动态化画面对象属性的另一种方法。要配置表达式，可使用一个附加选项卡。通过表达式，可以定义基于多个变量的条件，并通过条件改变多个属性。

- 当其中一个变量值发生变化时，将对表达式进行评估。
- 只要表达式的结果发生变化，属性就会随之改变。
- 如果所有条件都不返回 TRUE，则使用默认值。
- 如果定义了多个条件，则列表中第一个返回 TRUE 的条件会被使用。
- 无法求值的表达式将被跳过，例如由于语法错误或无法访问的变量。

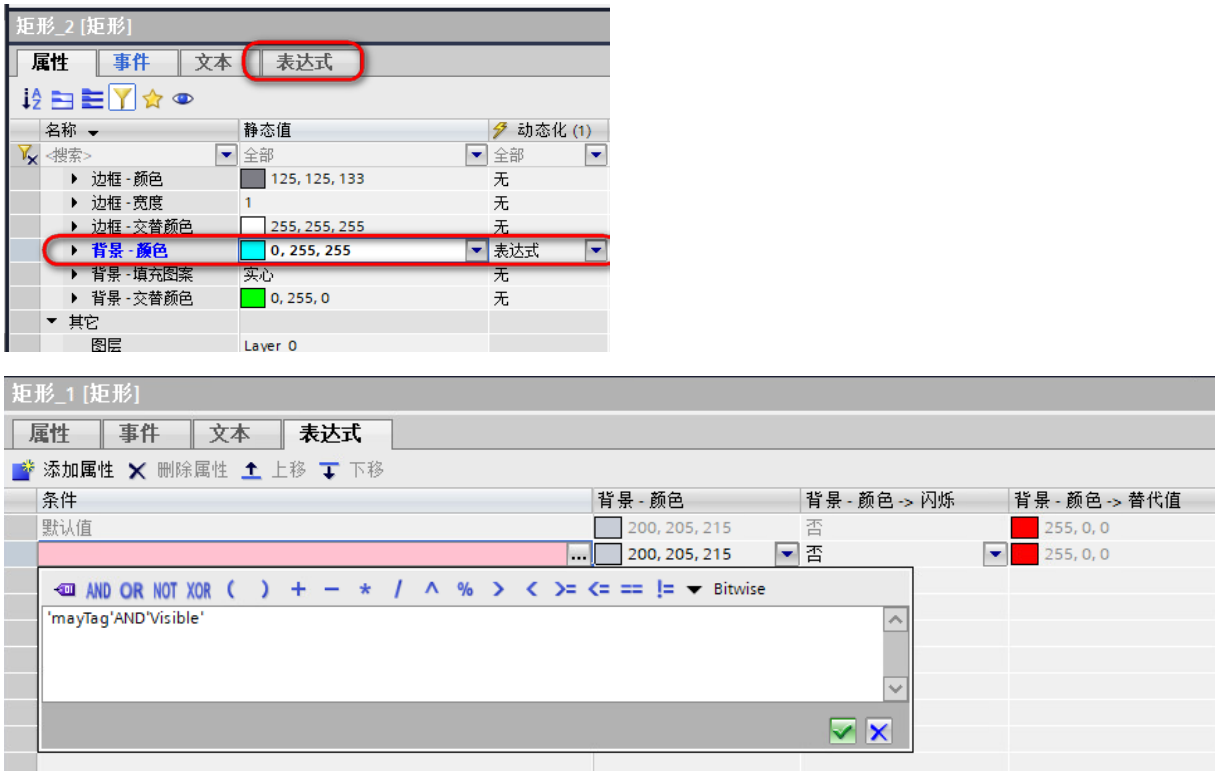


图3-42 作为动态化的表达式

注意表达式的定义顺序。这些表达式从上到下依次执行，一旦某个条件为真，其他条件将不再执行。在下面的示例中，顺序必须如下。如果定义的顺序相反，则 AND 条件永远不会被评估，因为 OR 条件总是先满足。

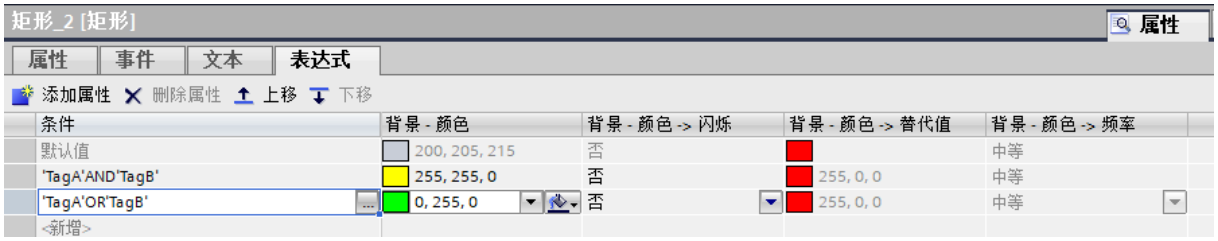


图3-43 表达执行顺序

3.2.3.1. 用例：根据单个位动态调整属性

定义

有些用例需要根据字中的单个位动态调整属性。以前，这必须通过脚本来实现，如下例所示。
根据位数返回不同的颜色值。

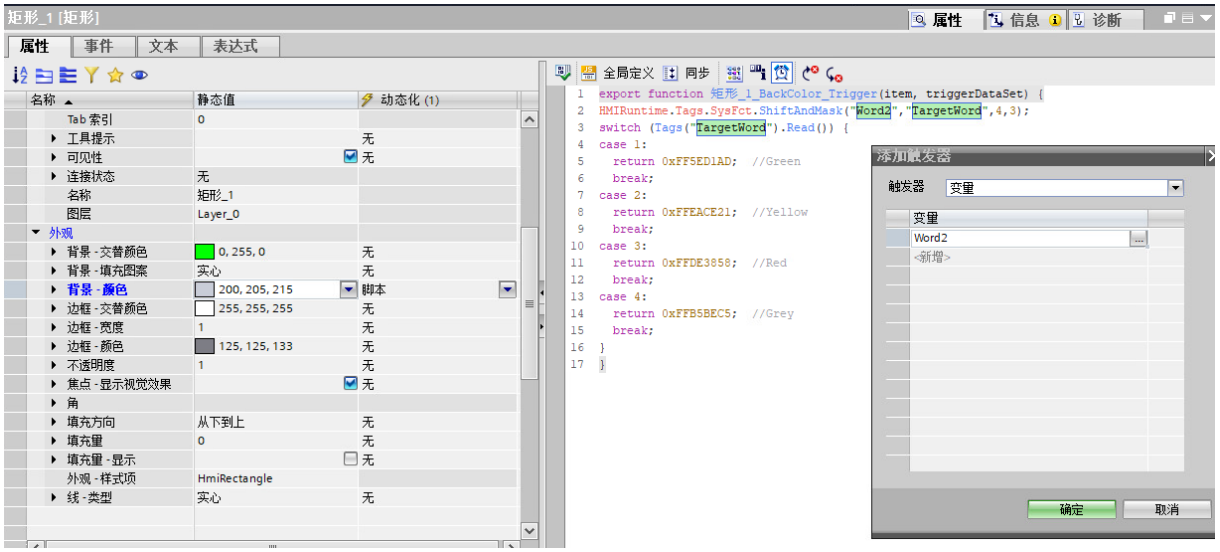


图 3-44 使用 ShiftAndMask() 根据特定位动态设置属性的脚本

解决方案

在表达式中使用位操作取代属性上的脚本动态化。SR8()- 运算符用于右移位，AND8()- 运算符用于屏蔽。

图 3-45 表达式属性

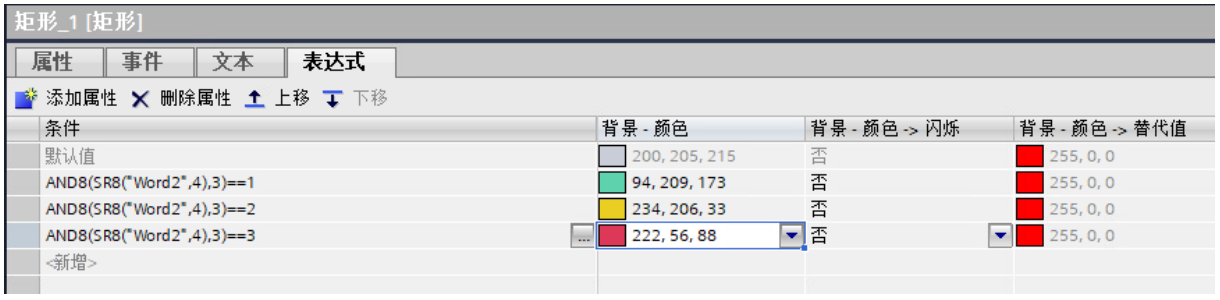


图 3-46 带移位和屏蔽运算符的表达式

3.2.4. 公式 V20

公式可用于通过变量动态化的任何数值属性（如背景颜色、可见性等等）。该属性无需使用任何脚本即可轻松动态化。使用公式可以定义基于多个变量的条件，如果使用的变量值中有一个发生变化，该公式就会被执行。公式的转换设置与表达式相同。

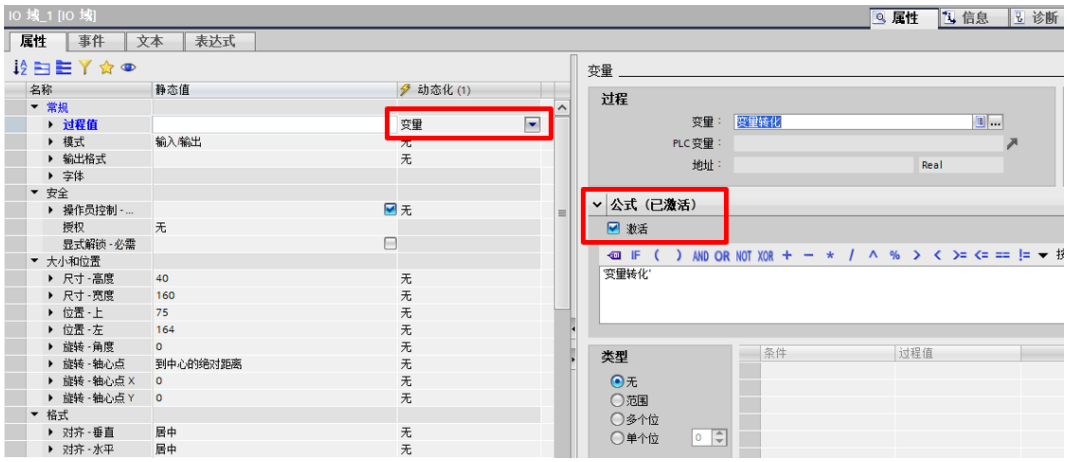


图3-47 基于公式的变量动态化

此外，系统还在公式中提供了一些转换函数。以转换不同单位组中的值或变量。
不同单位组中的数值或变量。使用转换函数比使用脚本更有效。

3.2.4.1. 用例：转换变量值实现动态化

定义
有些用例需要将一个变量转换成一个值（例如乘以 10）或将一个变量转换成另一个单位。
以前，这些都需要通过脚本来完成，如以下示例所示。

用例1：
变量乘以 10，然后根据乘数设置相应的矩形背景色。

Global definition | Synchronous

```
export function Rectangle_3_BackColor_Trigger(item, triggerDataSet) {
    var value;
    let tag = triggerDataSet('ConvertTag2').Value * 10;

    switch (tag) {
    case 10:
      value = HMIRuntime.Math.RGB(0, 255, 0, 255);
      break;
     case 20:
      value = HMIRuntime.Math.RGB(0, 0, 255, 255);
      break;
     case 30:
      value = HMIRuntime.Math.RGB(255, 255, 0, 255);
      break;
     case 40:
      value = HMIRuntime.Math.RGB(255, 0, 0, 255);
      break;
     case 50:
      value = HMIRuntime.Math.RGB(255, 0, 255, 255);
      break;
    default:
      value = HMIRuntime.Math.RGB(200, 205, 215, 255);
      break;
    }
    return value;
}
```

图 3-48 根据转换变量更改背景颜色的脚本

用例2：数值转换
以英里为单位的变量将转换为以公里为单位

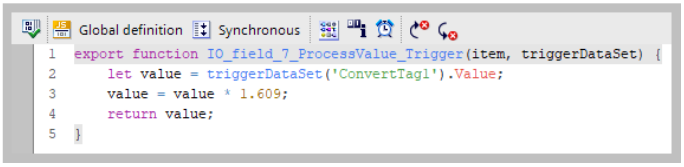


图 3-49 将英里转换为公里的脚本

解决方法

使用公式 V20 和提供的转换函数进行属性动态化，以取代脚本动态化。

用例1：公式表达式

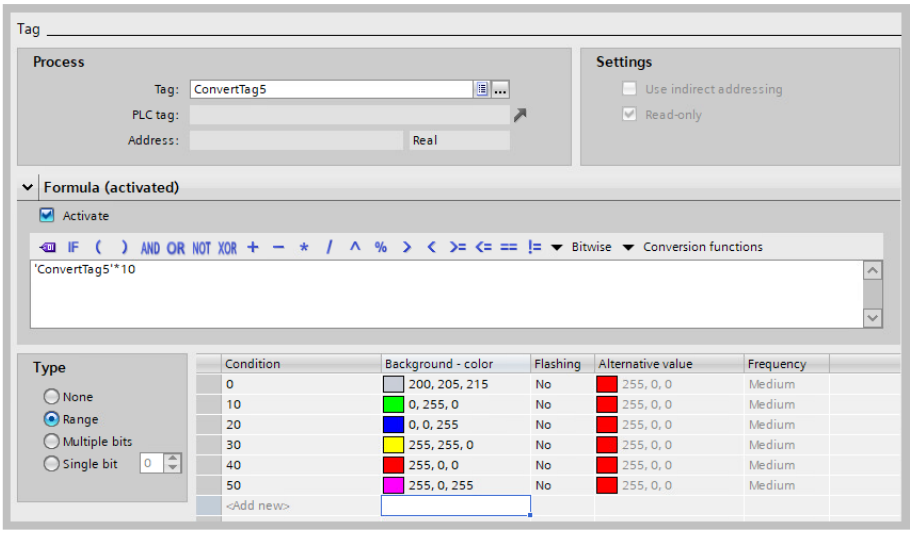


图 3-50 带有范围值的公式

用例2：数值转换

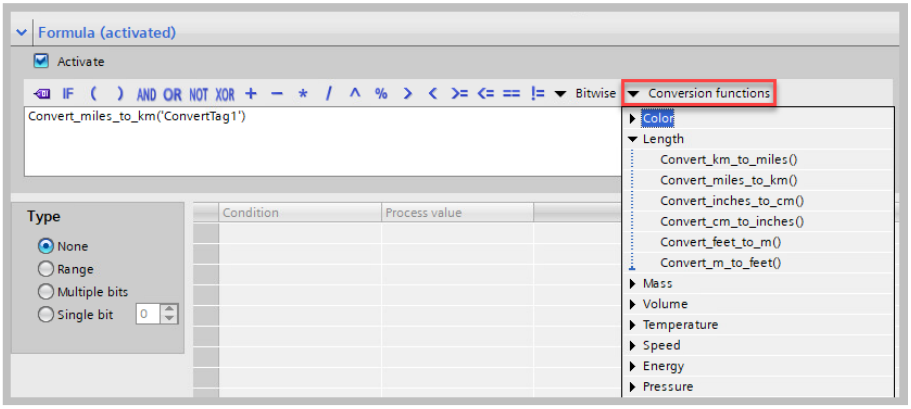


图 3-51 带转换功能的公式用法

3.2.5. 更多选项

根据属性的不同，也可以使用资源列表或闪烁作为动态效果。闪烁选项适用于颜色属性。资源列表适用于文本属性。

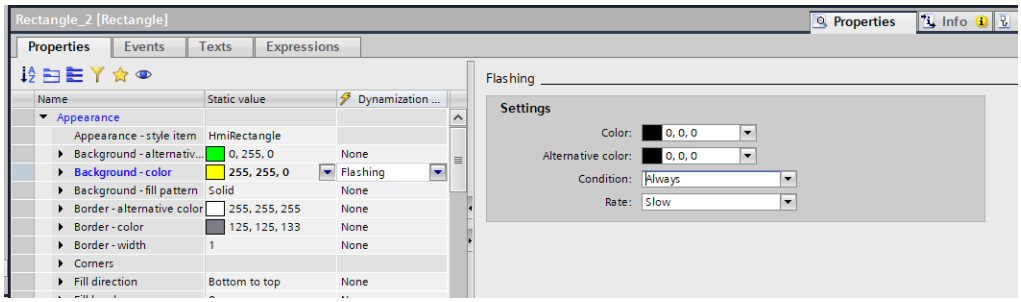


图 3-52 颜色属性的闪烁动态化

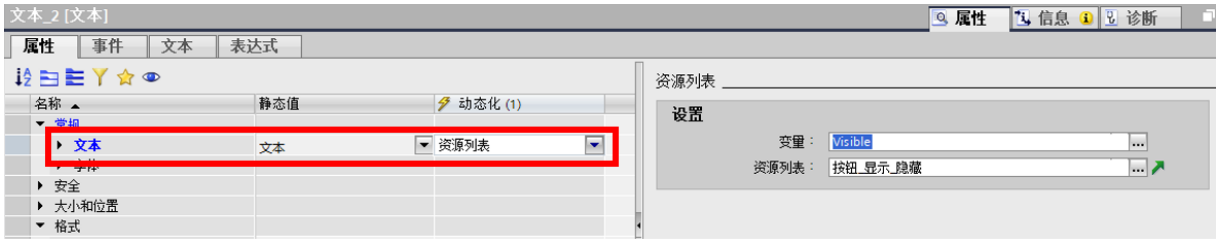


图 3-53 文本属性的资源列表动态化

3.2.5.1 使用案例： 将静态文本与文本属性的动态参数相结合

使用格式化文本是在按钮等文本属性中包含动态参数（变量或文本列表）的另一种选择。通过格式化文本，可以在画面对象的静态文本属性中将静态文本与来自变量或文本列表的动态值结合起来。

格式化文本还可与面板一起使用。面板无法在参数字段中使用文本列表。

这使得文本属性的动态化更容易，因为在这种情况下可以省略脚本。此外，它还可以将动态值输出和静态文本输出结合到一个对象中，例如带有相应单位的变量值。

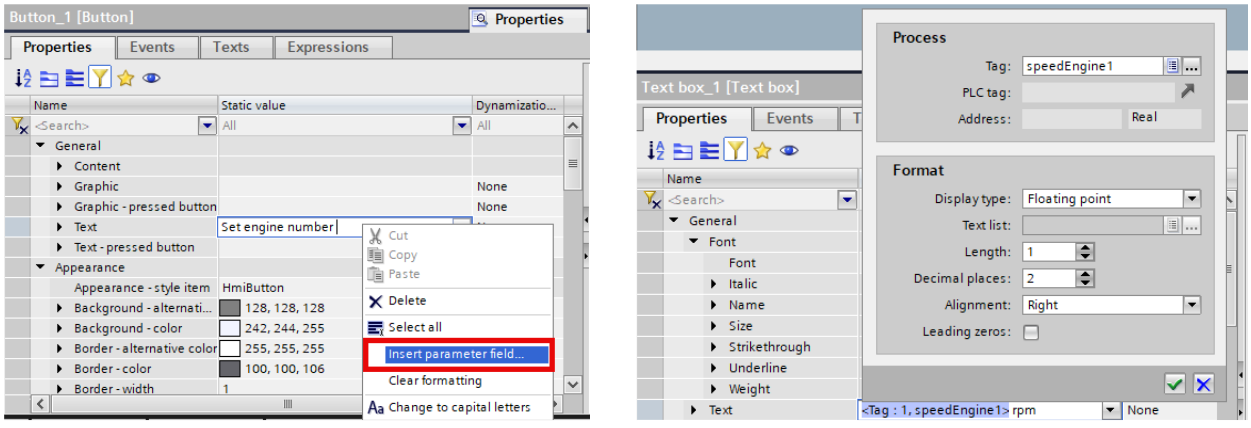


图 3-54 使用格式化文本的按钮文本动态化

动态化

尽可能使用变量动态化。如果需要更复杂的逻辑，则使用公式或表达式。尽量避免脚本动态化！只有在使用脚本可以避开使用多个画面对象时，才建议使用脚本（参见第 2.3.1 章）。

SIMATIC WinCC Unified - Tips and Tricks for Scripting (Java Script) 中详细介绍了不同触发器类型（包括周期触发、变量触发、报警触发...）的信息。

3.3. 面板的使用

面板用于项目的标准化。如果您想使用面板，请确保您对多个画面对象进行了配置。出于性能考虑，不建议对单个画面对象（如单个矩形）使用面板，因为面板接口会带来额外的消耗。在此用例中使用 WinCC Unified Corporate Designer（请参阅3.1.1.5章节使用案例：自定义样式 ）。

3.3.1. 面板使用的基本见解

使用 Faceplates 时，必须注意项目的一般注意事项，以实现最佳性能。
首先，要评估 Faceplates 与 V19 之前版本的兼容性，并探索使用更新功能重新实现的可能性。此外还要注意以下几点：

- 使用面板在尽可能高的版本（设备版本相同）
- 使用面板的画面对象进行复杂而广泛的逻辑配置
- 为面板接口选择合适的数据类型
- 在创建面板时，请注意画面的系统限制。7-12 英寸 Unified 精智面板的计算示例：
  7-12 英寸UCP 每个画面的对象数量: 800
  画面上的面板数量：20
  画面上的附加对象：20
  → 800 - 20 个画面对象- 20 个面板= 剩余760个对象可用于面板
  → 760 / 20 = 在此用例中每个面板可使用38个对象

3.3.1.1. 使用案例：Runtime 并非始终可见的面板

说明

只在需要时才显示内容的面板可以弹出式显示，也可以在工程的画面组态中配置，但设置为不可见。

解决方案

激活这些面板实例的可中止选项。通过此设置，当面板不可见时，由变量更改触发的循环脚本或脚本将不会执行。
不可见也指不在当前可见画面区域内的情况。

图 3-55 可悬挂面板属性

3.3.1.2. 使用案例：属性接口中的字符串

说明

在面板属性接口中，可以选择不同的数据类型。对于字符串来说，有两个选项：多语言文本和 WString（配置字符串）。
图 3-50 面板接口属性的数据类型

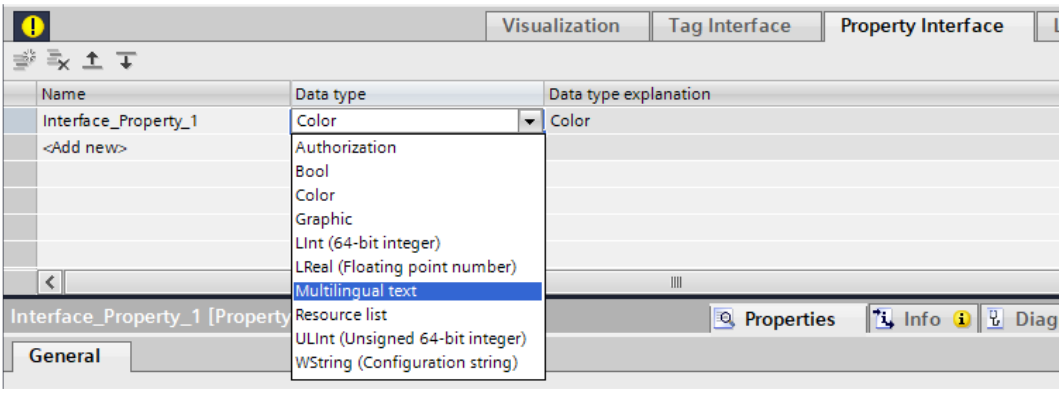


图 3-56 数据类型作为面板的属性接口

解决方案

如果需要用于面板类型内变量动态化的字符串，请选择多语言文本。配置字符串不能作为动态化链接到属性，只能通过脚本链接。

只有在传输配置设置（如趋势控制中的趋势或报警过滤器）时，才推荐使用配置字符串。

多语言字符串还可以保存不同运行语言的文本信息，并可在Runtime切换。

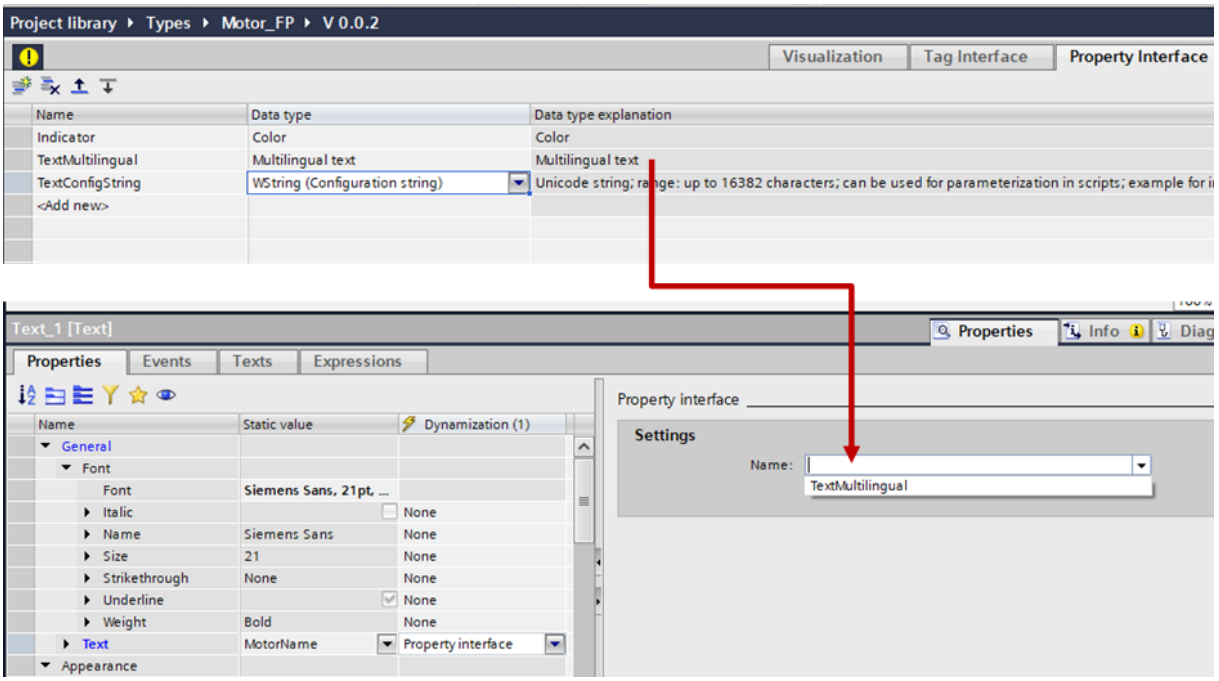


图 3-57 面板属性接口中的多语言文本和配置字符串

3.3.1.3. 使用案例：分层面板的动态数据连接

说明

面板可视化与 PLC 的标准化数据连接。虽然画面上可以同时显示多个面板实例及其各自的数据连接，但在许多情况下，一次只

需显示一个实例即可，而无需动态调整界面属性连接。

解决方案 1 - 使用系统函数动态更改界面 V18

在 V18 之前，必须用静态 HMI 变量连接面板界面的变量属性。要操作与相应变量的接口连接，可以使用系统函数 SetPropertyValue。

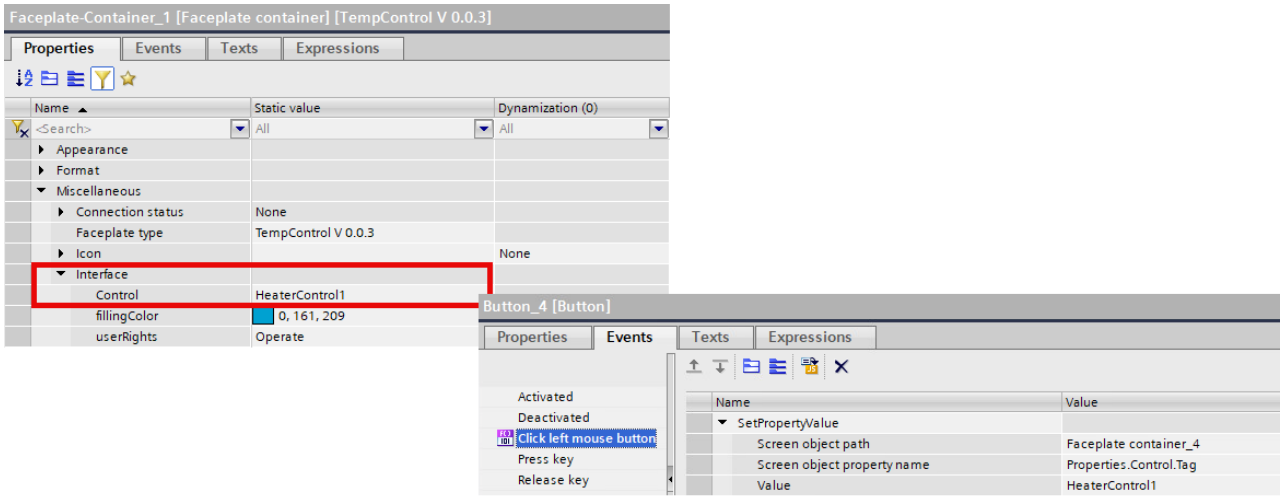


图 3-58 静态面板接口连接

解决方案2 - 使用变量前缀动态更改接口 V19

从 V19 开始，通过使用基于前缀的变量动态化，现在可以使用单个面板容器来显示不同的面板实例。

定义一个由静态部分和动态部分组成的变量名称作为变量参数。动态部分是变量引用，在Runtime 会被当前变量值替换。因此，生成的变量名称取决于变量引用的值。

如果要访问复杂的 PLC UDT，而这些 UDT 包含的信息远多于面板内部所需的信息，则尤其需要使用这种解决方案。将每个 UDT 作为独立的 HMI 变量实例加载，而不是使用多路复用。这样只会刷新所需的变量，而不会刷新整个 UDT 结构。有关使用多路复用的更多信息，请参阅链接 “使用案例 ”部分： 使用多路复用的 PLC UDT 阵列。

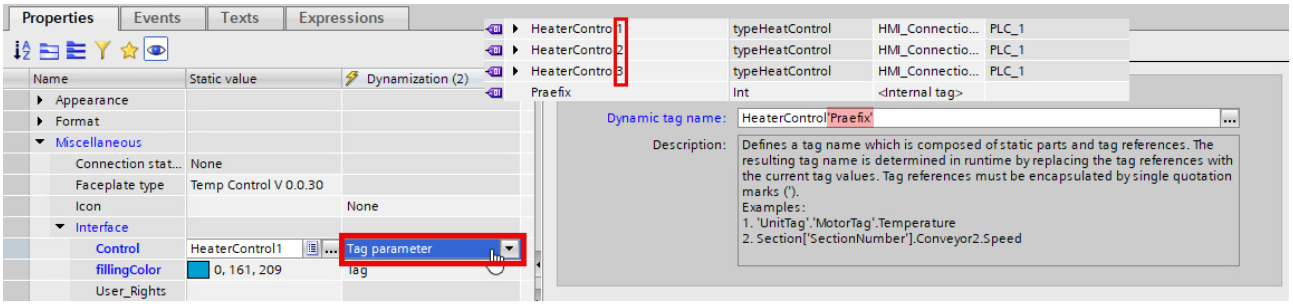


图 3-59 动态面板界面

要通过面板实例访问用户数据类型为 “typeControl ”的HMI 变量 “HeaterControl1 ”至 “HeaterControl4”，请定义另一个变量（例如 “Praefix”）作为 "HeaterControl'Praefix'界面属性的动态部分。根据 Runtime 中 “Praefix ”变量的值，HMI 变量 “HeaterControl1 ”至 “HeaterControl4 ”的变量名称将在面板界面中发生变化。

3.3.2. 面板中的文本列表

文本列表已配置好，现在要在面板中使用。由于面板不能直接访问该文本列表，下面将详细说明如何处理这种情况。

在面板中集成文本列表有多种可能的解决方案，各有利弊。至关重要的是，首先要明确在面板中使用文本列表的目的。

分析：

首先，确定文本列表中实际要在面板中使用的文本数量，以及项目中使用的面板实例数量非常重要。

如果要传输到面板上的文本元素数量较少，则有一个最佳解决方案。在这种情况下，在面板属性接口中使用单个变量是最佳解决方案。有关该程序的更多信息，请参阅以下内容：用例： 传输文本列表的子集

如果需要从文本列表中使用的文本较多，则需要对配置进行额外的考虑。在这种情况下，为文本站点的每个文本元素设置单独变量的解决方案并不是最佳解决方案，因为随着文本元素和面板实例的增加，工程工作量也会显著增加。合适的解决方案取决于具体的使用情况。

必须检查同一面板的不同面板实例所使用的文本列表是否不同。因此，必须检查是否必须在面板上动态或静态配置文本列表。 用例：文本列表的静态应用**错误!未找到引用源。**

用例：静态使用文本列表

面板的所有实例都要使用标准化文本列表。

文本列表是所有面板实例的基本框架。

文本列表可作进一步调整，因此所有面板实例的文本必须能快速、方便地调整。

用例：文本列表的动态应用

面板的各个实例需要不同的文本列表。
面板实例的使用案例由引用的文本列表及其引用的接口变量决定。

3.3.2.1 使用案例：传输文本列表子集

描述：

本用例涉及从配置的文本列表中将单个或少量文本元素传输到面板。这种情况可能出现在设计文本列表时，而每个面板实例只需要其中的几个元素。本用例的一个可能应用场景是使用带有两个按钮的面板来控制一个执行器。

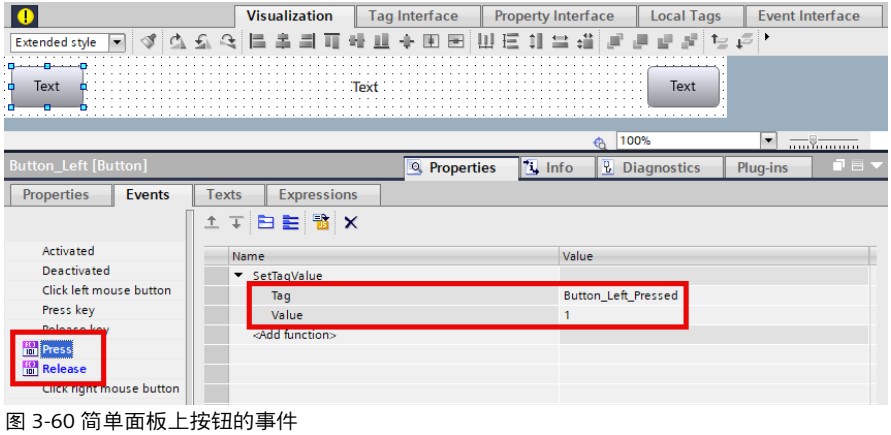


图 3-60 简单面板上按钮的事件

按钮会将变量设置为 true，然后在释放时将其重置为 false。

按下这两个按钮可在 “真 ”和 “假 ”之间切换两个布尔变量。按下这些按钮所触发的操作由 PLC 代码决定。这种灵活性使得同一个面板可用于多个执行器，如气缸或电机，按钮变量和面板标题也会相应调整。

该列表包括系统中打算通过面板控制的所有执行机构。

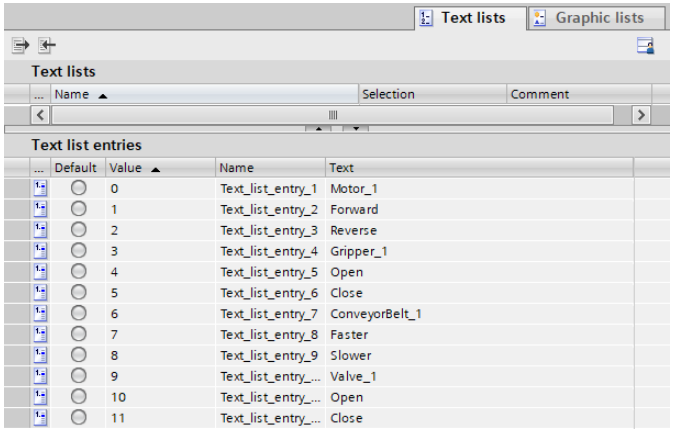


图 3-61 执行机构控制面板变量文本列表示例

在这种情况下，需要创建一个面板实例来控制电机。从配置的文本列表中，现在只需要相应执行器的文本元素：

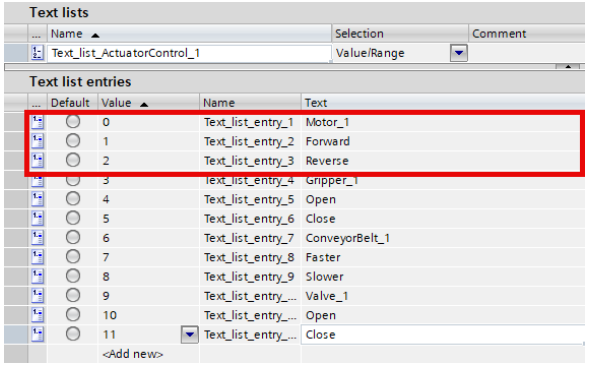


图 3-62 简单面板的文本元素

解决方案

在此过程中，文本列表中要传输到**面板**的单个条目将作为多语言文本传输。因此，必须在**面板**界面配置三个多语言文本变量。

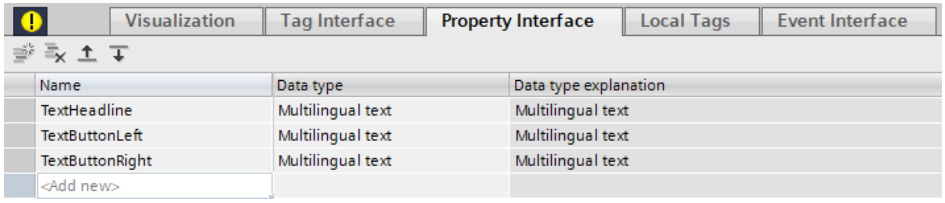


图 3-63 简单面板的属性界面

弹出窗口中的面板在弹出窗口中打开面板时，无法在界面上引用多语言文本对象。多语言文本对象，因此

不适合这种使用情况。

然后就可以在对象的所需属性中直接引用多语言文本。这里不需要使用脚本

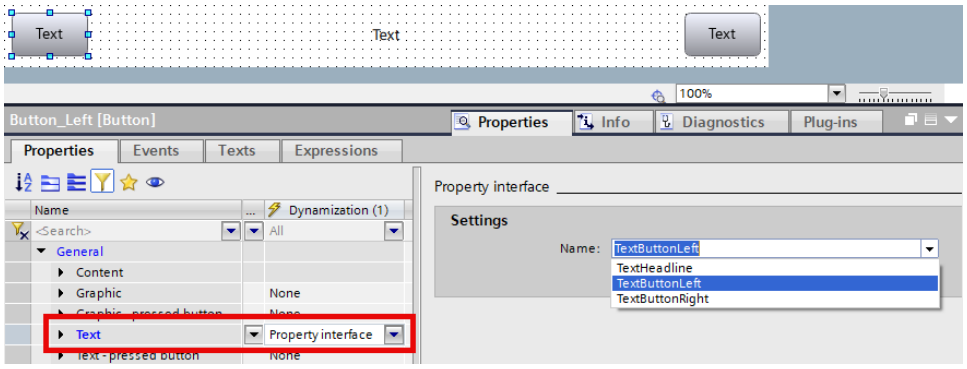


图 3-64 简单面板上按钮的文本属性

此处的 “Text ”属性引用了左侧按钮变量的多语言文本。

通过在界面中指定索引，可以在实例的**面板**容器上定义所需的文本列表文本元素。这些索引可以在不同的**面板**实例中使用，从而实现文本元素的动态使用。

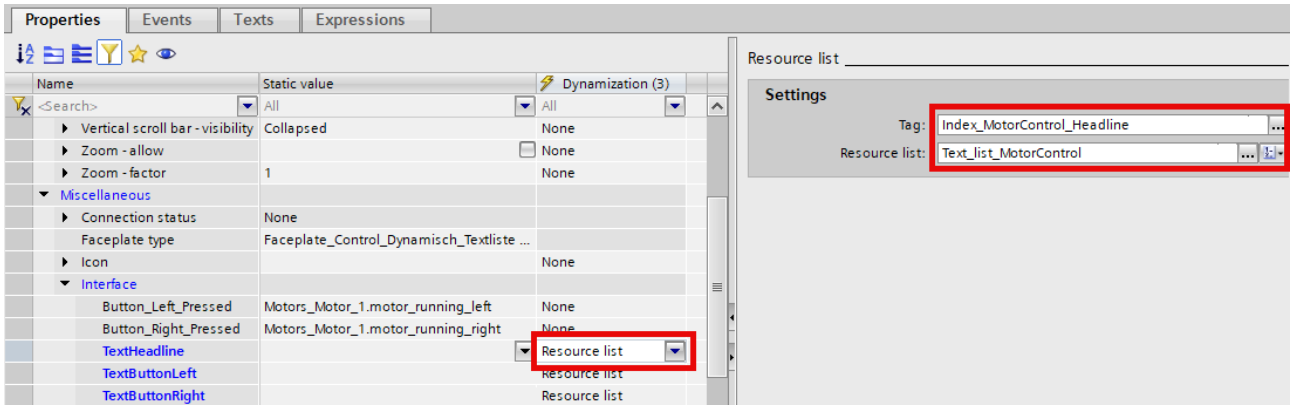


图 3-65 简单面板上的面板容器界面

这是包含接口中所有引用的面板容器。

通过控制执行机构（本例中为电机）的接口引用的变量可按如下方式配置。然后在 PLC 代码中进行解释。

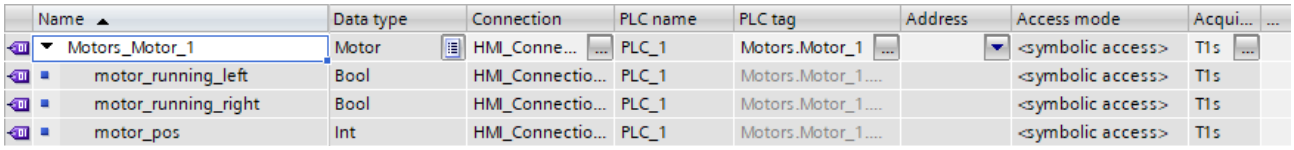


图 3-66 简易面板 PLC-UDT

优势：

- 无需编写脚本
- 动态

3.3.2.2. 使用案例：静态使用文本列表

说明

本案例涉及文本列表的静态使用，这些列表用于面板的所有面板实例，无需在面板容器上进行动态调整。

一种典型的情况是显示系统组件（如执行器）的状态。每个面板实例必须使用相同的文本列表，以确保执行器状态显示的一致性。对执行器状态/状态的任何更改或添加都应自动应用于该面板的所有面板实例。

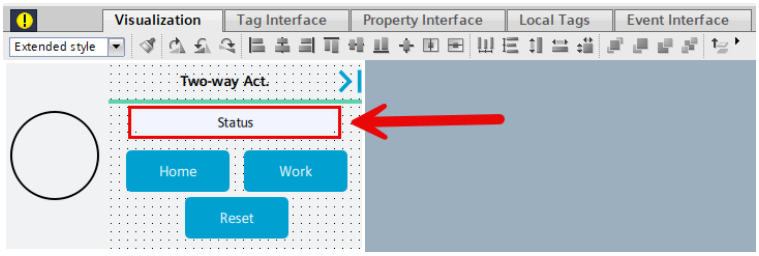


图 3-67 带有显示执行机构状态文本框的双向执行机构面板

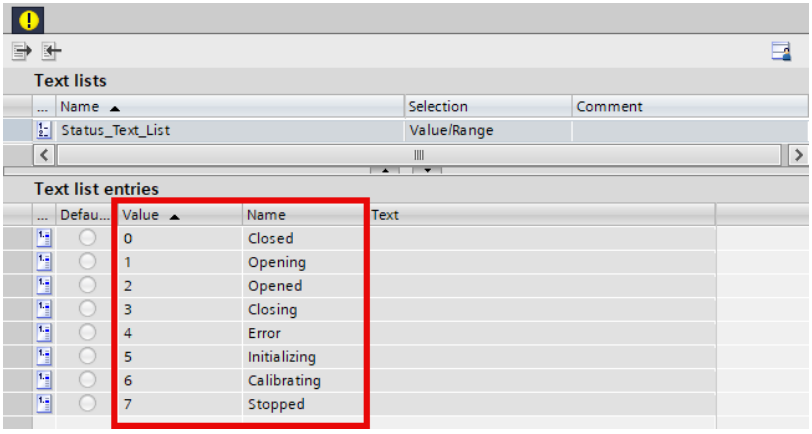


图 3-68 反映执行机构可能状态的配置文本列表

解决方案：文本列表类型 V19

对于文本列表的静态使用，我们建议使用文本列表类型，该类型在第 19 版中引入，自第 19 版更新 2 起可在对象属性中直接引用。首先必须在库中为所需的文本列表创建相应的文本列表类型。

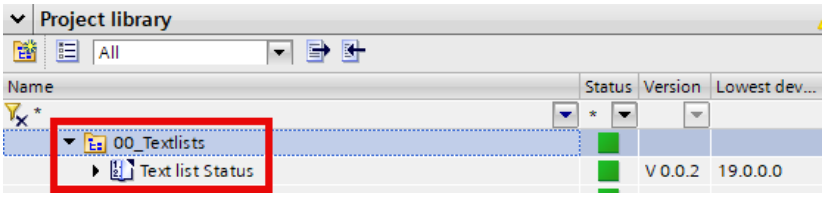


图 3-69 库中的文本列表类型

由于面板可以直接与库通信，因此使用文本列表类型可以直接访问文本列表。这样，所需的文本元素就可以直接引用到面板中对象的相应属性。

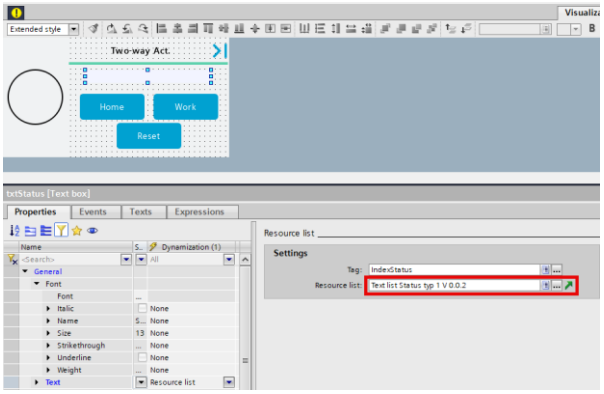


图 3-70 双向执行器面板上状态文本框的文本属性

在这里，库中配置的文本列表被引用到文本框的“文本”属性中。文本列表的索引可以根据执行机构的状态进行设置。

优势：

工程设计无需在界面中指定文本列表，从而在引用面板容器时最大限度地减少时间和潜在错误。

3.3.2.3. 使用案例：文本列表的动态使用

说明

要创建一个面板，其中每个面板实例都要使用不同的文本列表。因此，要根据面板的使用情况调整文本列表。

这里的情况与用例：传输文本列表子集 中的情况相同，不同之处在于需要控制的执行器数量。该面板旨在为工厂的系统部件创建一个标准化的控制元件。系统部件的图形用户界面结构保持不变，只有 PLC 中的逻辑不同，因此按钮的变量也不同。

它现在可以控制四个执行器，而不仅仅是一个。

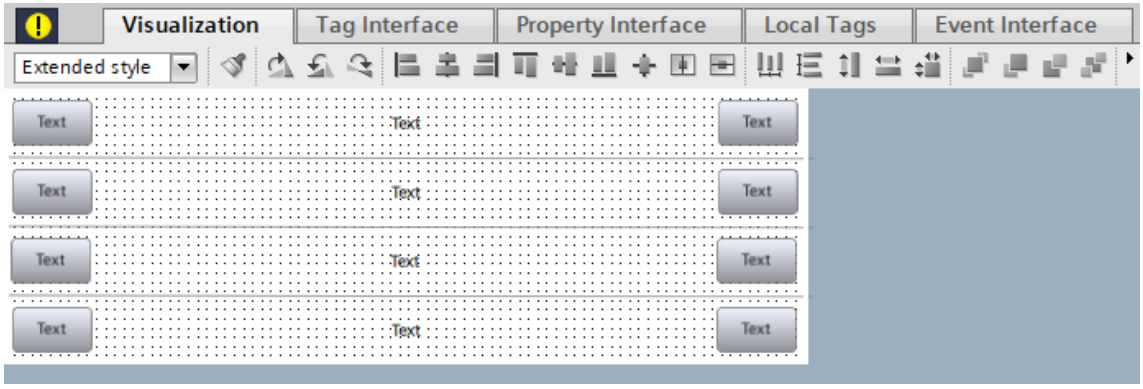


图 3-71 带四个控制单元的扩展面板

在这种情况下，使用的文本列表与 “使用案例：传输文本列表的子集 ”中的相同。但现在使用的是该文本列表的全部文本内容。

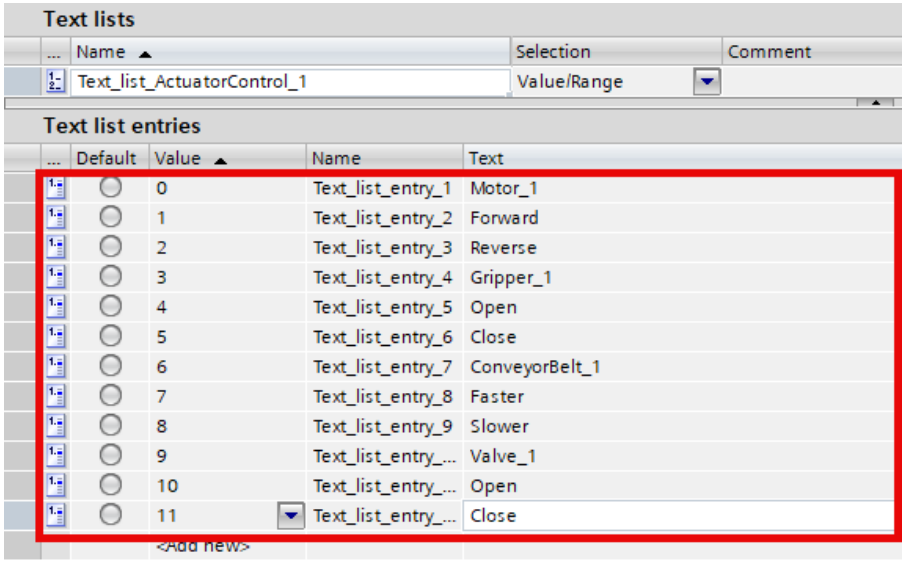


图 3-72 执行机构控制示例文本列表所需的文本元素

解决方案：资源列表

在这种情况下，可以使用面板属性界面中的资源列表将文本列表动态转移到面板中。使用该解决方案，可根据预期用途（如附件部分）将所需的文本列表绑定到面板容器上，然后使用脚本将文本传输到对象的属性中。
要访问面板中的文本列表，必须首先在属性界面中指定该列表。

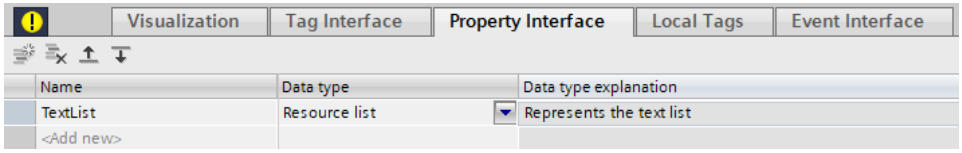


图 3-73 扩展面板的属性界面

然后就可以读出该资源列表，并将其转移到对象的属性中，为此应使用 “TextsByValues ”方法。使用该方法，可以快速、轻松地读出所需的条目并将其存储到数组中。在这种解决方案中，无法在对象级别直接引用文本列表元素，多语言文本就是这种情况。

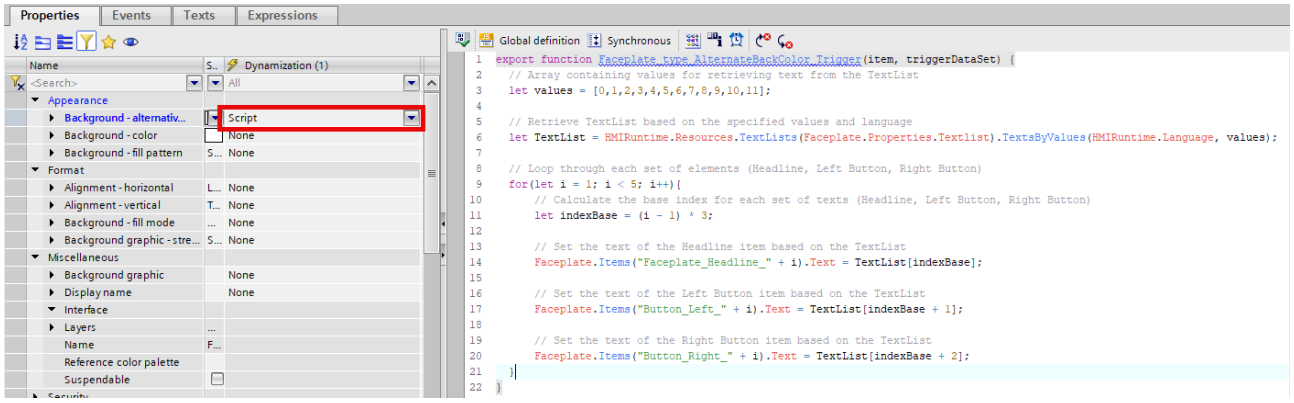


图 3-74 扩展面板的加载事件

在这种情况下，选择了面板类型的属性动态化，其中触发变量“@CurrentLanguage ”用于在语言更改时重新加载文本列表。

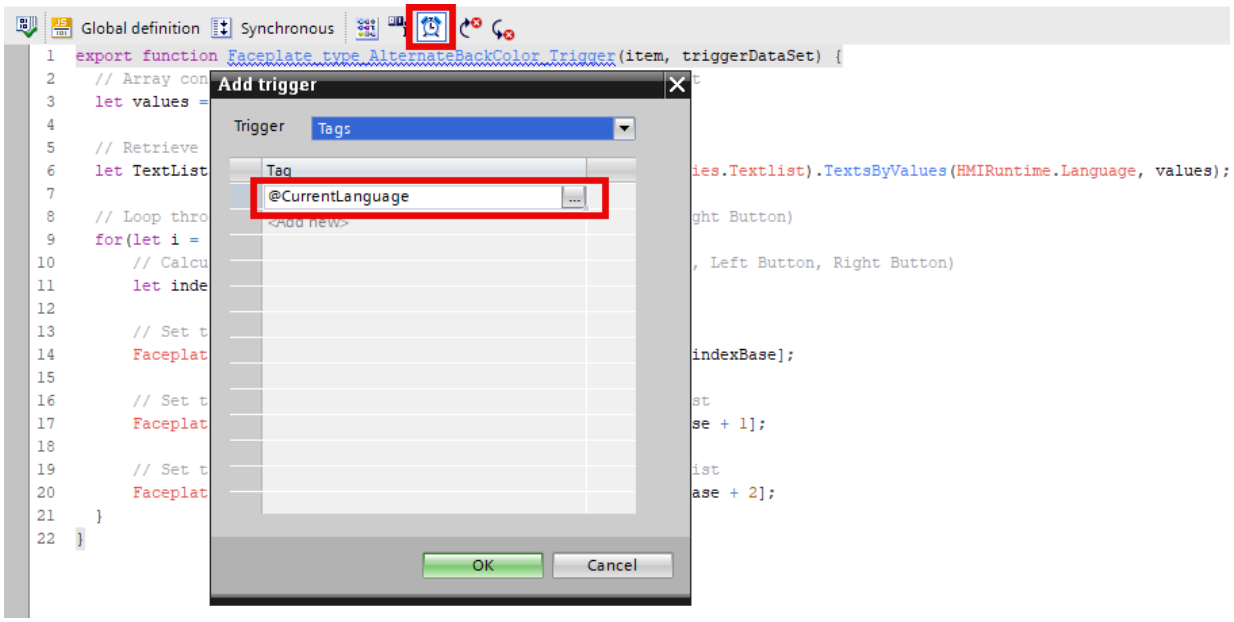


图 3-75 用于更改文本列表的动态脚本

注意	明确的为对象命名，以便于用 for 循环遍历这些对象，例如调整文本属性。

现在可以在画面中创建该面板类型的面板实例，然后将所需的文本列表传输到面板上。

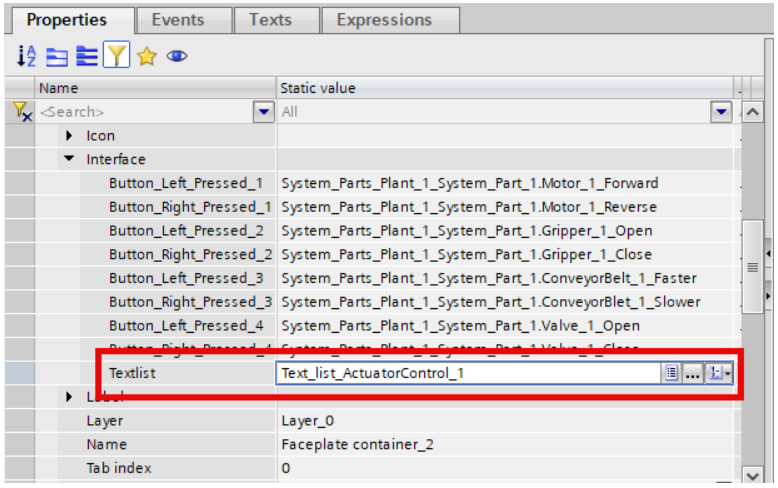
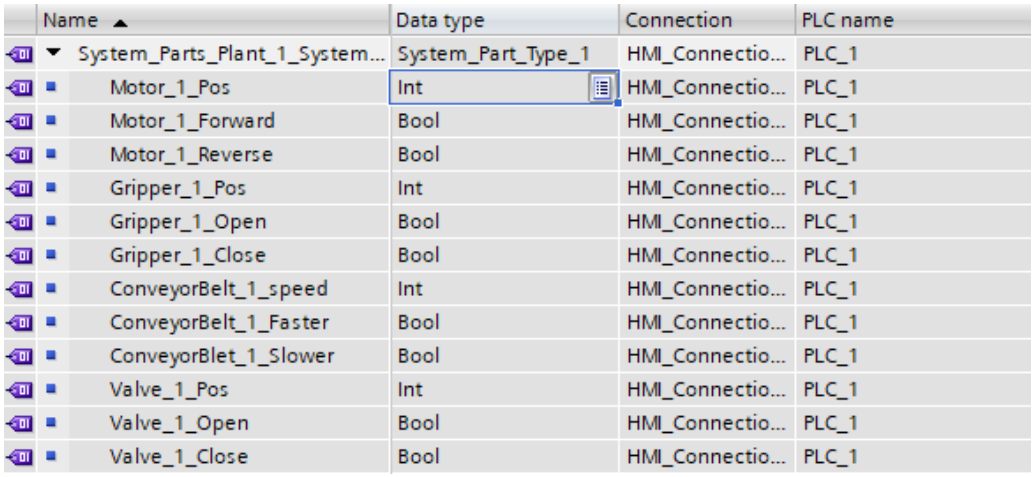


图 3-76 扩展面板上的面板容器接口

在该面板中，首先引用上图所示的文本列表，然后在面板中读取。其次，这里引用了控制执行器所需的变量，然后在 PLC 中进行处理。



Name	Data type	Connection	PLC name
System_Parts_Plant_1_System...	System_Part_Type_1	HMI_Connectio...	PLC_1
Motor_1_Pos	Int	HMI_Connectio...	PLC_1
Motor_1_Forward	Bool	HMI_Connectio...	PLC_1
Motor_1_Reverse	Bool	HMI_Connectio...	PLC_1
Gripper_1_Pos	Int	HMI_Connectio...	PLC_1
Gripper_1_Open	Bool	HMI_Connectio...	PLC_1
Gripper_1_Close	Bool	HMI_Connectio...	PLC_1
ConveyorBelt_1_speed	Int	HMI_Connectio...	PLC_1
ConveyorBelt_1_Faster	Bool	HMI_Connectio...	PLC_1
ConveyorBlet_1_Slower	Bool	HMI_Connectio...	PLC_1
Valve_1_Pos	Int	HMI_Connectio...	PLC_1
Valve_1_Open	Bool	HMI_Connectio...	PLC_1
Valve_1_Close	Bool	HMI_Connectio...	PLC_1

图 3-77 扩展面板 PLC-UDT

优势：

- 易于配置
- 传输大量文本元素时性能最佳

3.4. 脚本的使用

3.4.1. 脚本触发器

避免使用循环触发器，而应使用变量触发器。如果仍然需要循环脚本，并且使用情况允许（例如，同步数据或与数据库进行数据交换时），则在计划任务中配置脚本。与在画面中配置脚本相比，计划任务是在后台的一个单独进程中运行，对项目造成的负荷较小。

但也要注意，在使用"变量-自动"设置时，所有在脚本引用的变量都会被添加。这可能会在变量值写入时引起大量触发器触发和无限循环。

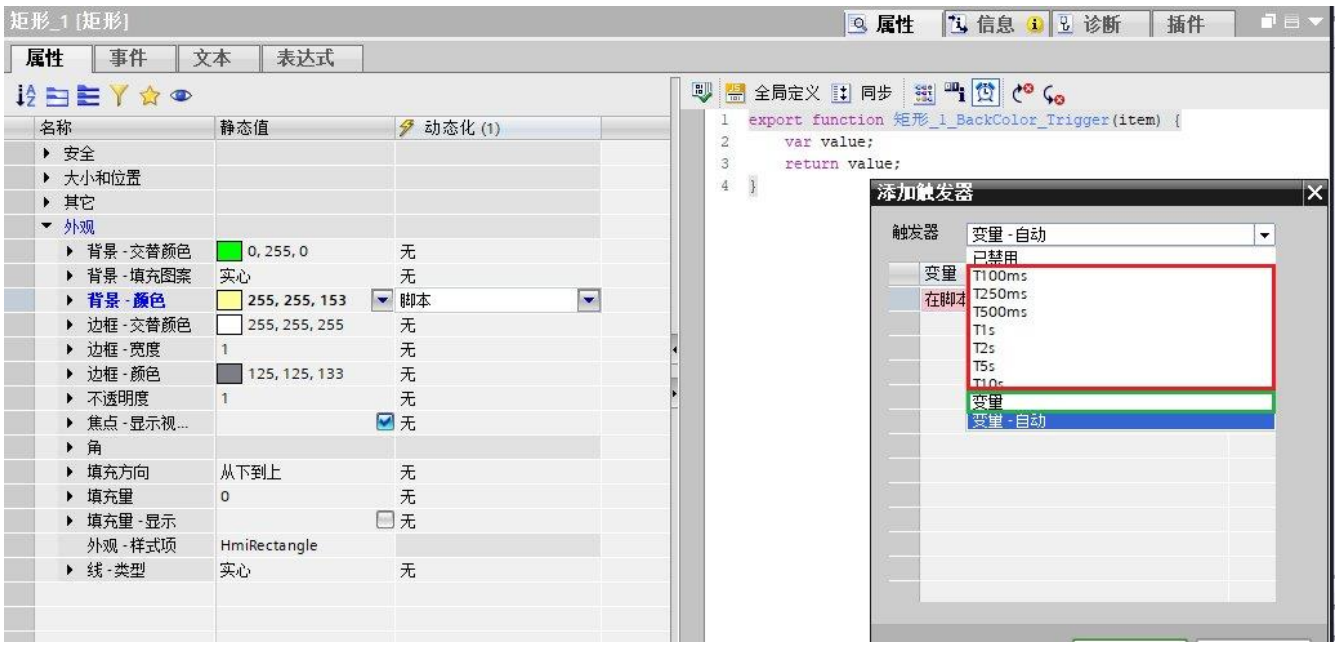


图3-78 脚本动态化的脚本触发器

如果两个脚本由同一个变量触发，则将它们合并为一个脚本。

脚本触发

使用变量触发，避免使用循环触发

3.4.1.1. 用例：不同的脚本使用相同变量触发

定义

当需要使用脚本动态化时，有些脚本可能会被同一个变量触发。因此，多个脚本会由同一个变量触发。

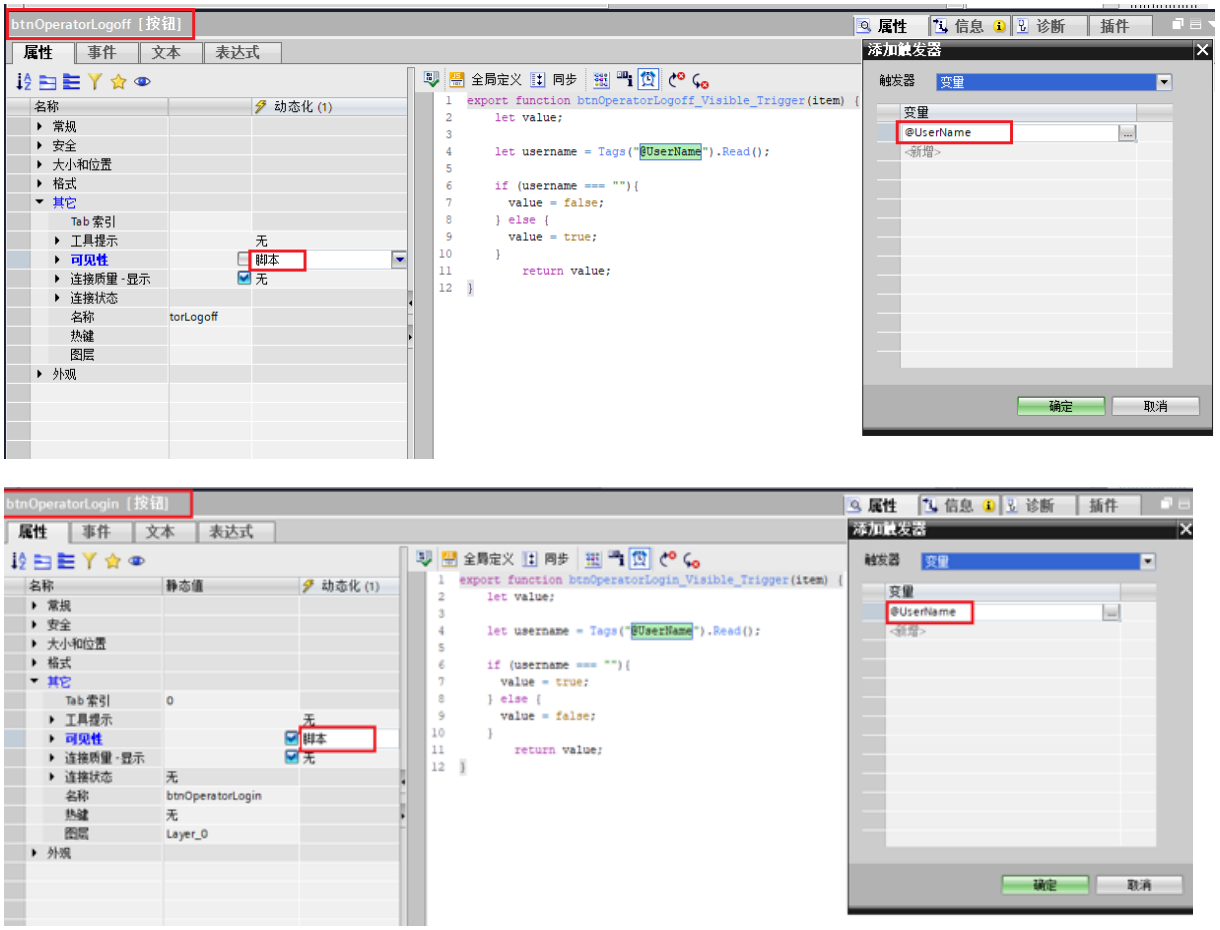


图3-79 使用相同触发变量的脚本动态化

解决方案1

在画面加载事件中为此变量创建一个订阅，并使用一个脚本集中管理属性更改。目前已经有一个用于变量订阅的代码段。尤其是在脚本中执行相同的脚本逻辑（如if-else）时，通过一个订阅可以节省大量的脚本执行时间。

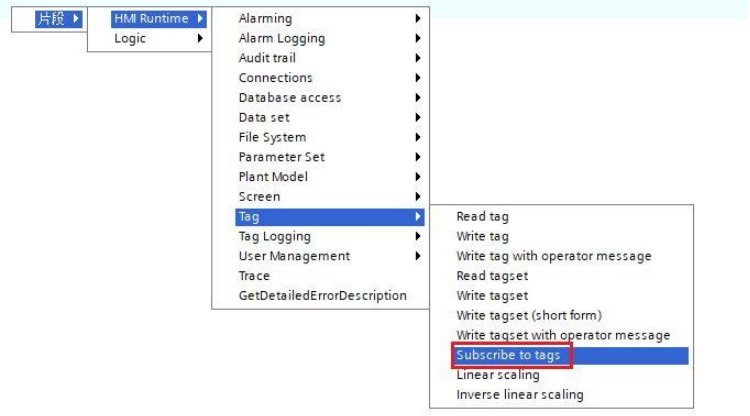


图3-80 变量订阅片段

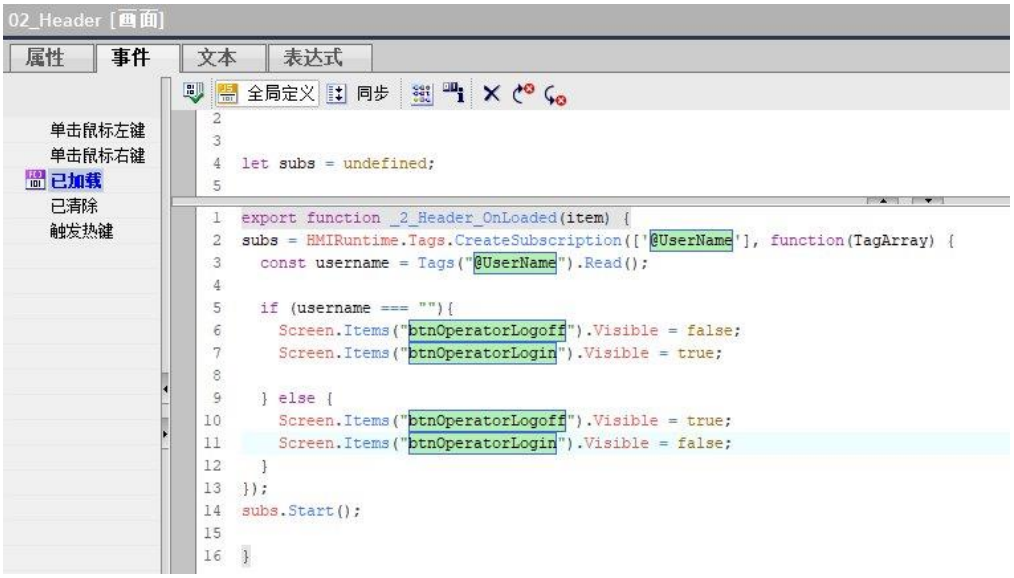


图3-81 已加载事件画面中的变量订阅

解决方案 2

在画面属性上创建脚本，用一个脚本集中管理属性更改。将脚本的触发器设置为应导致属性更改的变量。通过“triggerDataSet ”访问脚本中的触发器变量值。

triggerDataSet.Value 返回变量值，无需再次读取变量，从而提高了性能。

建议将此脚本集中添加到画面的属性中，而不是添加到单个画面对象中。这里使用了 “背景 - 备选颜色 “属性，因为它很少用于其他动态效果。

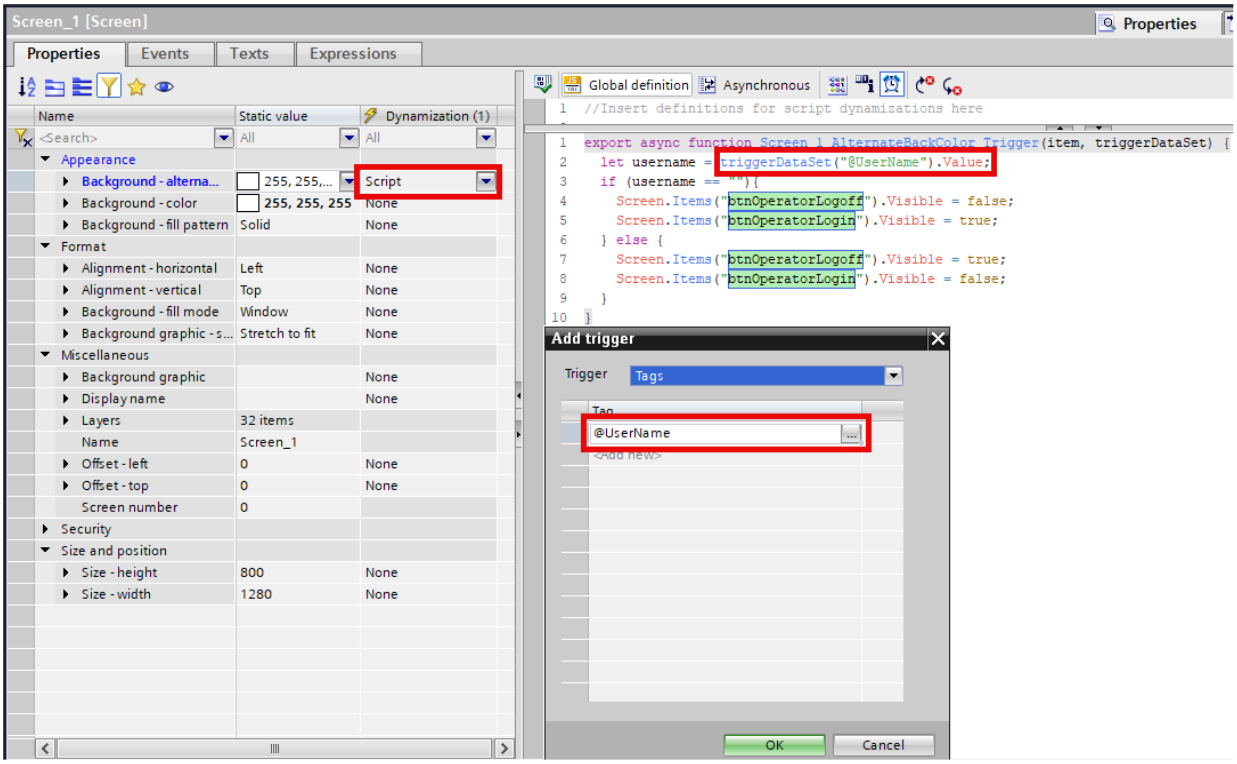


图 3-82 用于更改图像属性的 "triggerDataSet

triggerDataSet V19 Upd2

要访问变量触发器属性，请使用 triggerDataSet

3.4.1.2. 使用案例：与画面对象属性无关的触发脚本

说明

当需要根据变量变化触发脚本，且脚本与画面对象属性无关（没有项目需要调整）时，有很多选择来放置该脚本。

可以使用范围外的对象触发脚本，这些对象不会直接显示在画面内容上。当脚本与画面上的任何对象无关时，就可以使用这种方法。

在这个示例中，三个超出范围的矩形是通过 “背景 - 备选颜色 “属性的脚本动态设置的。当变量触发脚本时，属性不会发生任何变化，但会调用一个函数。(见图 3-83）

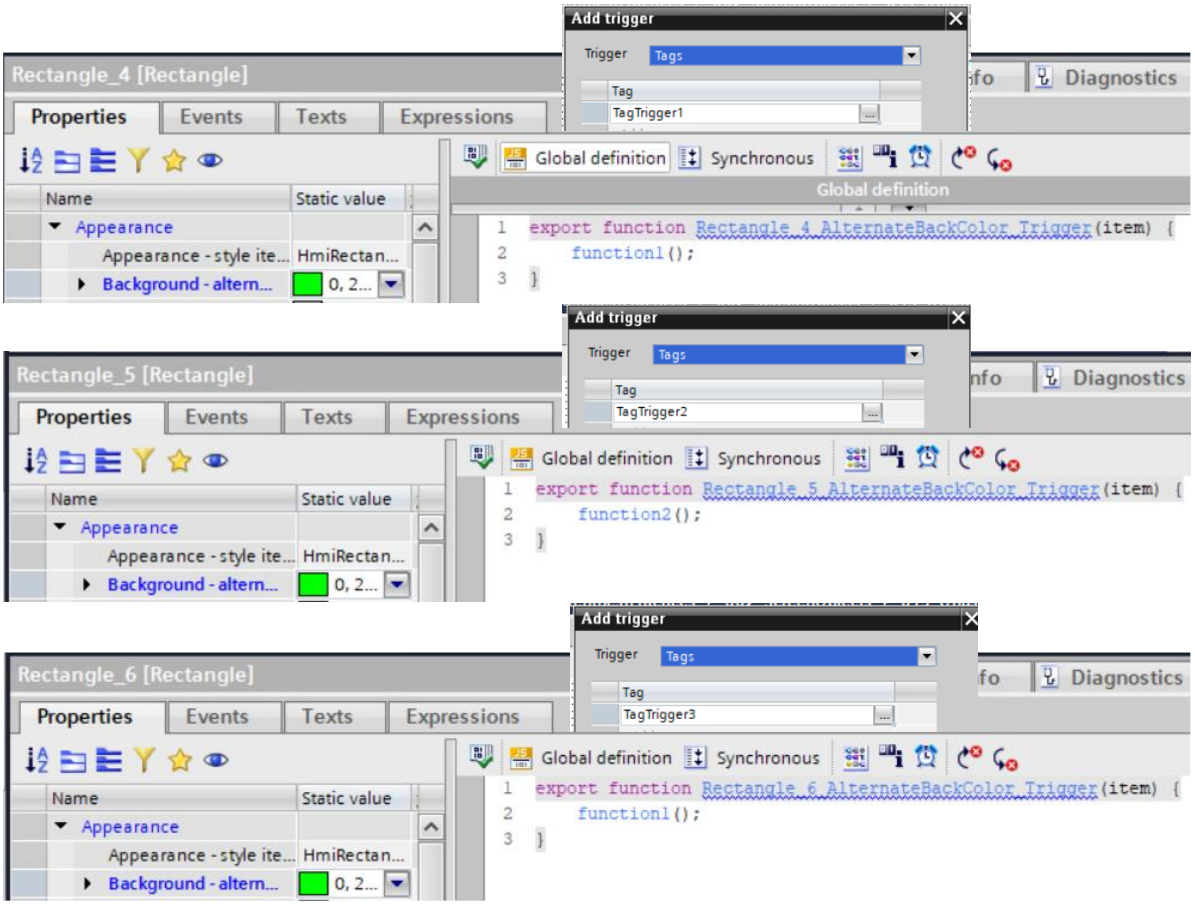


图 3-83 带有未使用画面项目属性的触发脚本

解决方案1

为此，可以使用计划任务，而不是使用超出范围的对象。这样就可以删除不可见和实际未使用的对象。由于该解决方案仅适用于不引用画面对象的代码，因此在下文中将针对该用例介绍第二种解决方案。

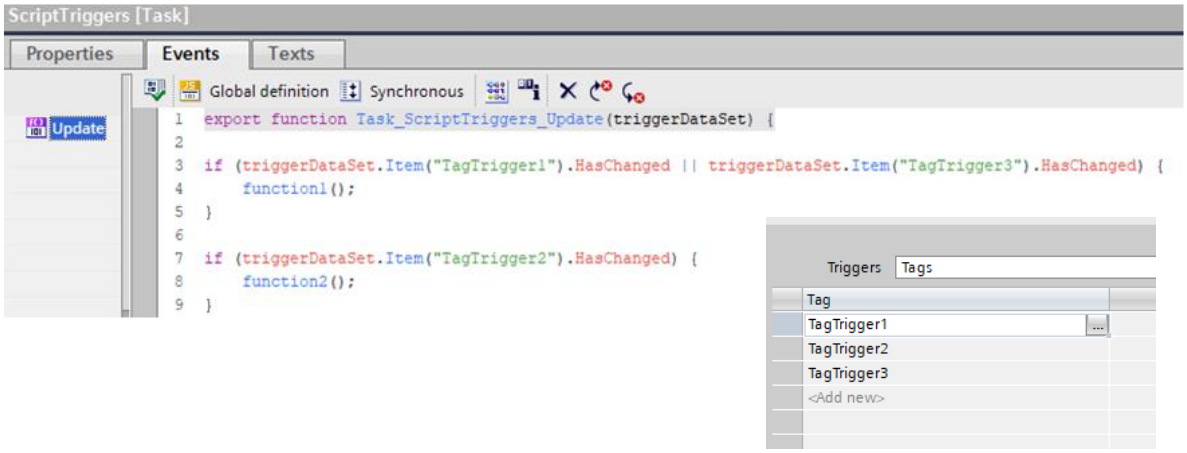


图 3-84 变量触发代码的计划任务

解决方案2 V19 Upd2

为此，可以使用画面属性，而不是使用超出范围的对象。这样，就可以删除不可见和实际未使用的对象，并对画面进行集中配置。

更好的办法是使用画面属性 “Background - alternative color”，并在其中添加三个变量触发器。要确定哪个变量被触发，可以使用系统函数 “triggerDataSet()”。有了这些信息，就可以调用相应的函数（见图 3-85）。这样就可以删除三个超出范围的矩形。

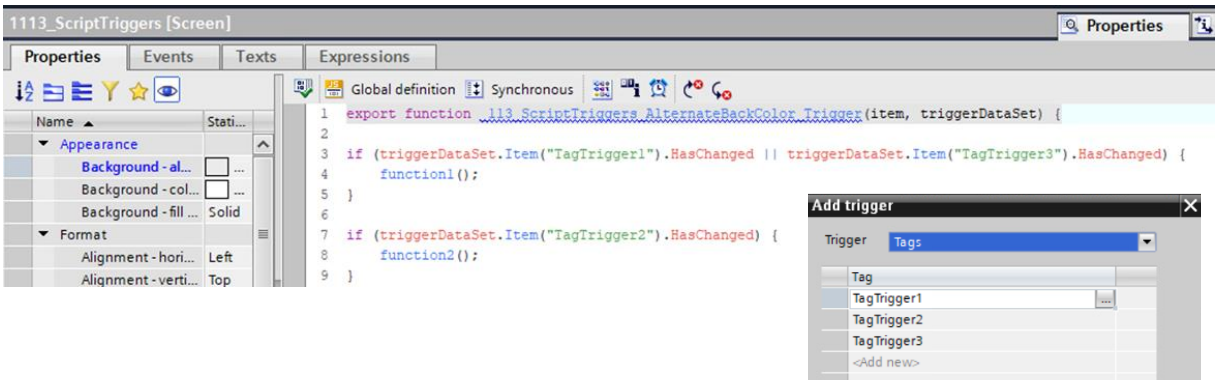


图 3-85 利用画面属性集中触发脚本

3.4.2. 高效编程

高效编程就是要减少脚本中可能导致问题的部分，甚至是对脚本结果没有影响的部分。需要注意的要点很多。因此，需要检查代码是否包含以下列表中的元素：

- 删除全局定义区域中未调用的函数或导入声明
- 首选使用系统函数列表中的系统函数，而不是在JavaScript 编辑器中编写脚本
- 删除空事件

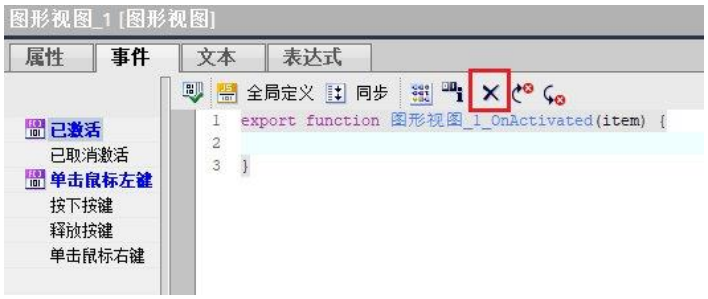


图3-86 删除空事件

- 有针对性地使用 break 语句，避免不必要的循环迭代

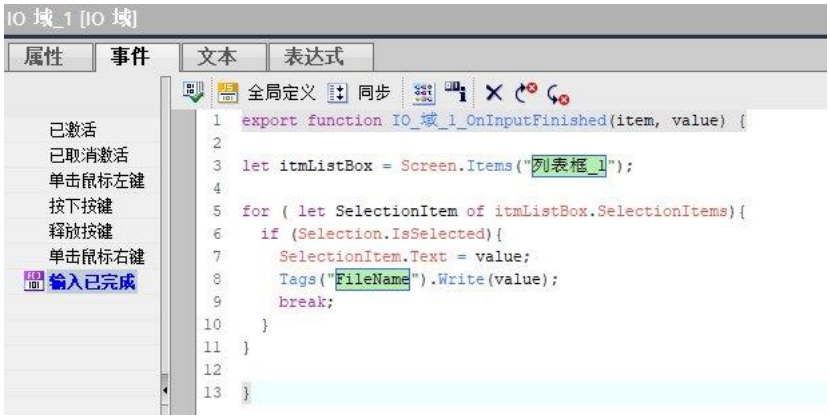


图3-87 提前跳出 for 循环的Break 语句

- 删除或注释不再使用的常量、变量定义和调试跟踪语句
- 只读取脚本中使用的变量
- 如果两个脚本由同一个变量触发，则将它们合并为一个脚本
- 读取变量一次，将其保存在一个中间变量中，如果脚本中多次使用该变量，则重复使用该中间变量
- 尽可能使用"const "定义（如果值不需要在脚本上下文中更改），否则就用"let "定义变量。不要再使用"var"，因为它已是不建议使用的 java 脚本标准
- 使用switch-case 代替if-else

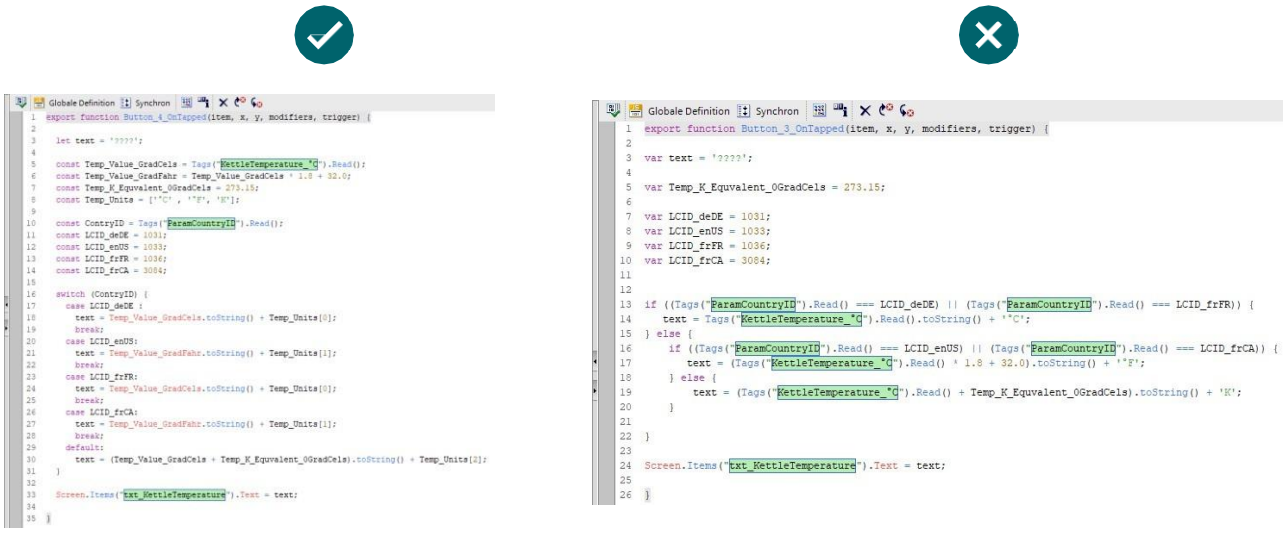


图3-88 Switch case与if-else

尽可能使用异步脚本。对于需要较长时间的调用，如数据库访问，只提供异步。对于变量访问，支持同步和异步模式，因为变量访问通常更快。有关脚本调用的更多信息，请参阅Java 脚本技巧和窍门。

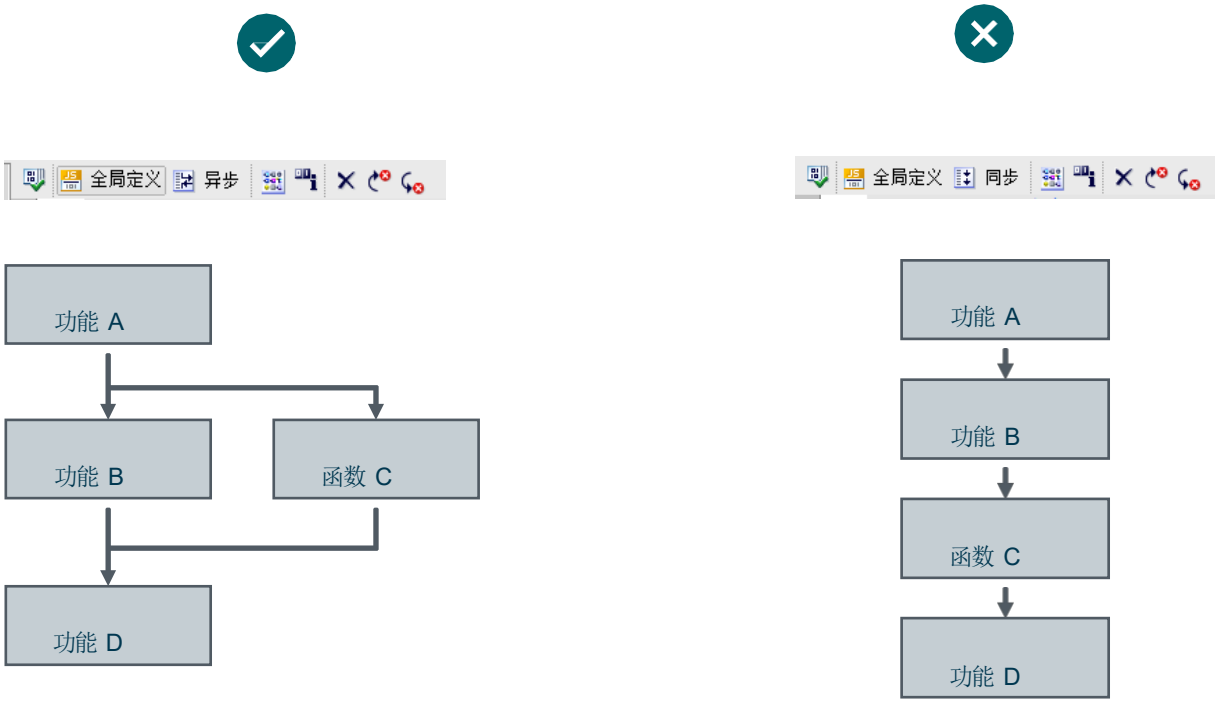


图3-89 异步脚本调用与同步脚本调用的对比

- 请注意脚本的执行频率，尤其是在画面加载过程中（见图 2-9）。下面的示例是在列表框过程值属性的更改事件中执行的脚本。这些属性的更改事件的可见性可通过眼睛图标在工程中控制是否隐藏 V19 。 .

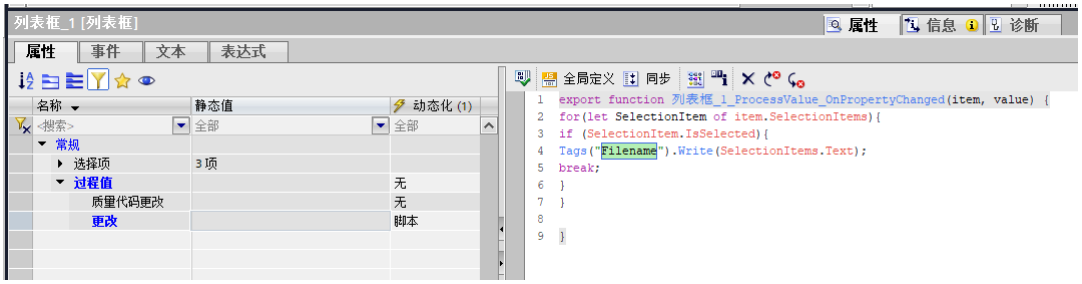


图3-90: 列表框过程值更改的事件

- 通过JS脚本一次性读取几个文本列表条目

  HMIRuntime.Recources.TextList(txtList).TextsByValue(HMIRuntime.Language, value); 而不是使用 for 循环。

```
let value = []
for(let i = 1; i >= count; i++)
{
  values.push( i + currentPage * 6);
}
let texts = HMIRuntime.Resources.TextLists(txtList).TextsByValue(HMIRuntime.Language, values);
}
```

图 3-91 一次性读取测试列表条目

3.4.2.1. 使用案例：写入和读取多个变量

说明

为了写入和读取单个变量，提供了大量系统函数例如 SetTagValue()，还可以通过 Tags("Tagname").Write("value") 或 Tags("Tagname").Read() 等脚本进行寻址。

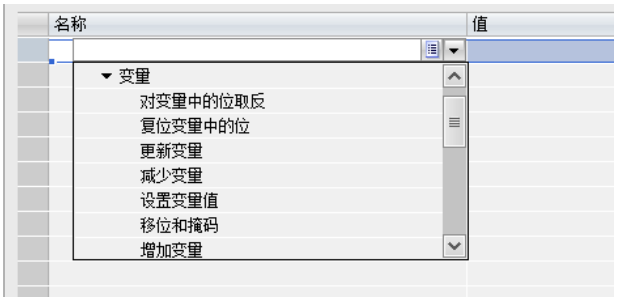


图3-92: 变量部分的系统函数节选

例如：

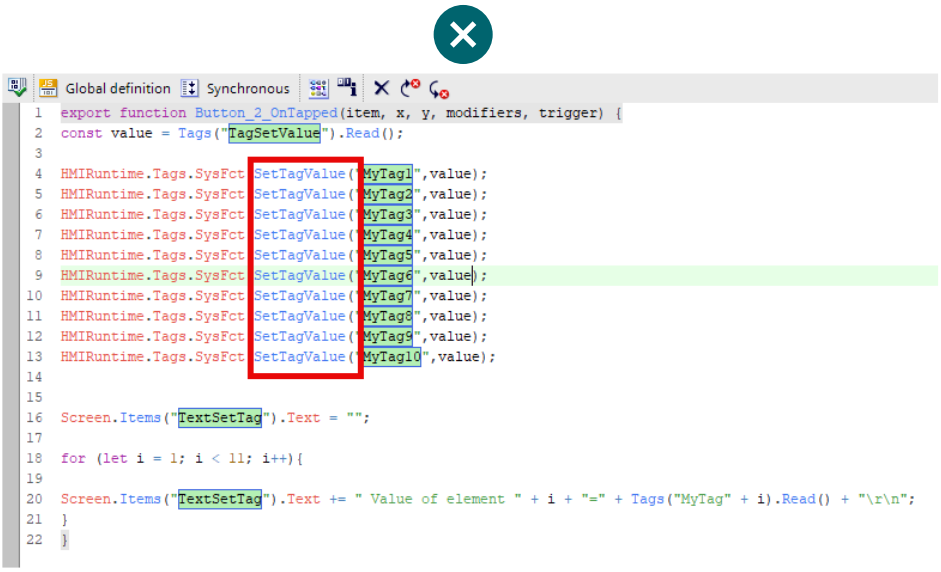


图 3-93 SetTagValue 的多次调用

但也可以创建一个变量集，涵盖一个容器中的多个变量。

解决方案

在写入或读取多个变量时创建一个变量集，并使用其写入和读取方法只需要向PLC 发送一个汇总命令。

下面的截图显示了使用 TagSet 功能对脚本进行的自定义。

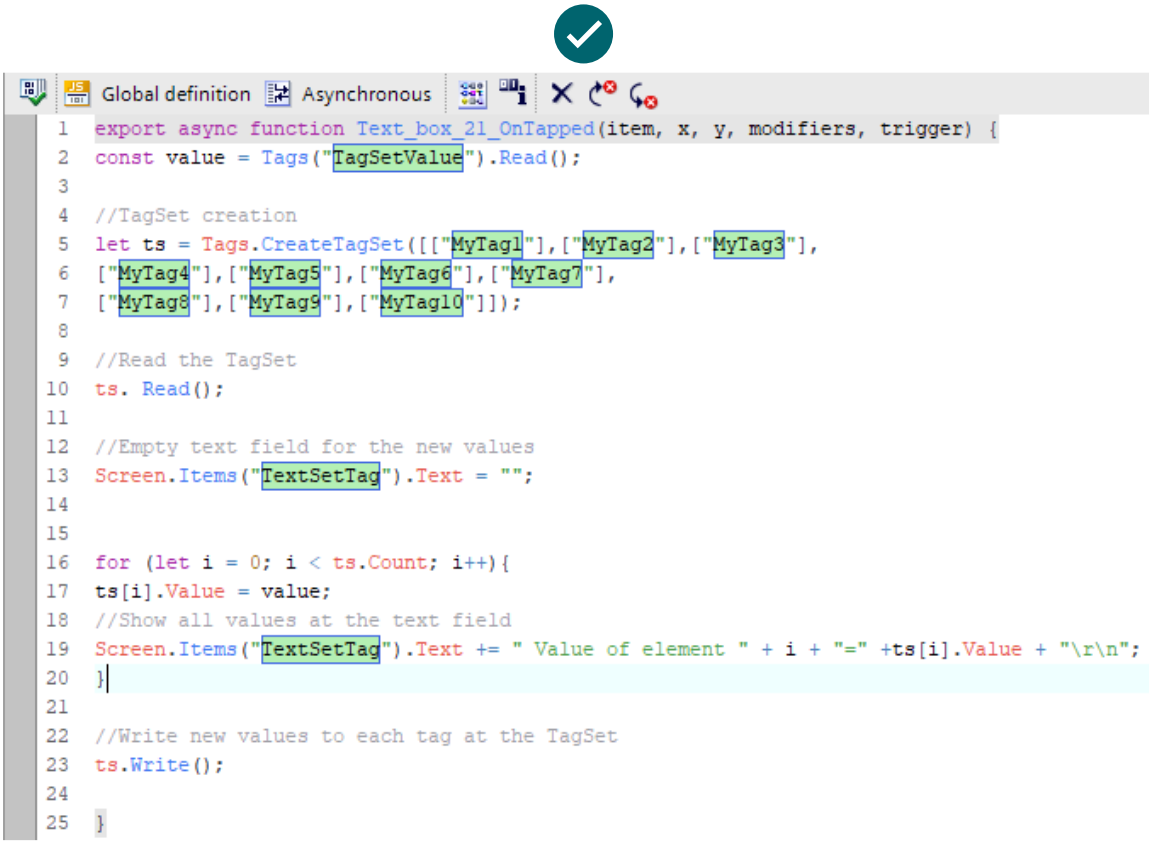


图 3-94 使用一个变量集读取多个变量

读写变量

使用 TagSet 读写多个变量，而不是单次重复调用 SetTagValue。尤其适用于同步读写多个控制变量。

有关脚本的更多信息，请阅读 SIMATIC WinCC Unified - Tips and Tricks for Scripting (Java Script)。

3.4.2.2. 计算密集型脚本的使用

说明

某些脚本方法需要较大的系统内存，被认为是计算密集型的。应仔细考虑项目中的使用情况，并将其减少到最低限度，以提高项目的总体性能。因此，计算密集型方法不应在加载事件中使用，也不应被循环触发器调用。最好只在需要时使用这些方法，这可以通过在按钮的点击事件中使用这些方法或在计划任务中使用这些方法来实现，因为计划任务有自己的脚本上下文（参见 2.3）。

这些脚本编写方法被认为是计算密集型的：

- 审计： ReadElectronicRecord()
- 报警： CreateSubscription(), GetActiveAlarms()
- 报警记录： HMIRuntime.AlarmLogging.Read()
- 文件系统： ReadFile(), Browse()
- 变量记录： Read()

3.4.2.3. 使用案例：访问画面对象

定义

通过脚本访问画面对象，通常有两种方法：

- Screen.Items(« 对象引用 »)
- Screen.FindItem(« 对象路径 »)

工程师根据实际项目需要选择合适的方法。

解决方案1：访问同一个画面中的画面对象

“Items”的返回值是返回当前画面中所有的画面对象的列表。您可以通过索引或变量名访问画面对象列表。因此，当画面对象与脚本位于同一个画面时，可以选择使用Screen.Items()而非Screen.FindItem()。

解决方案2：访问另一个画面中的画面对象

可以通过使用“FindItem”方法结合画面对象的路径访问另一个画面中的画面对象。您可以通过相对或绝对路径访问整个项目中的所有的画面对象。所以，仅当需要访问不同画面中的画面对象时才需要使用Screen.FindItem()方法。

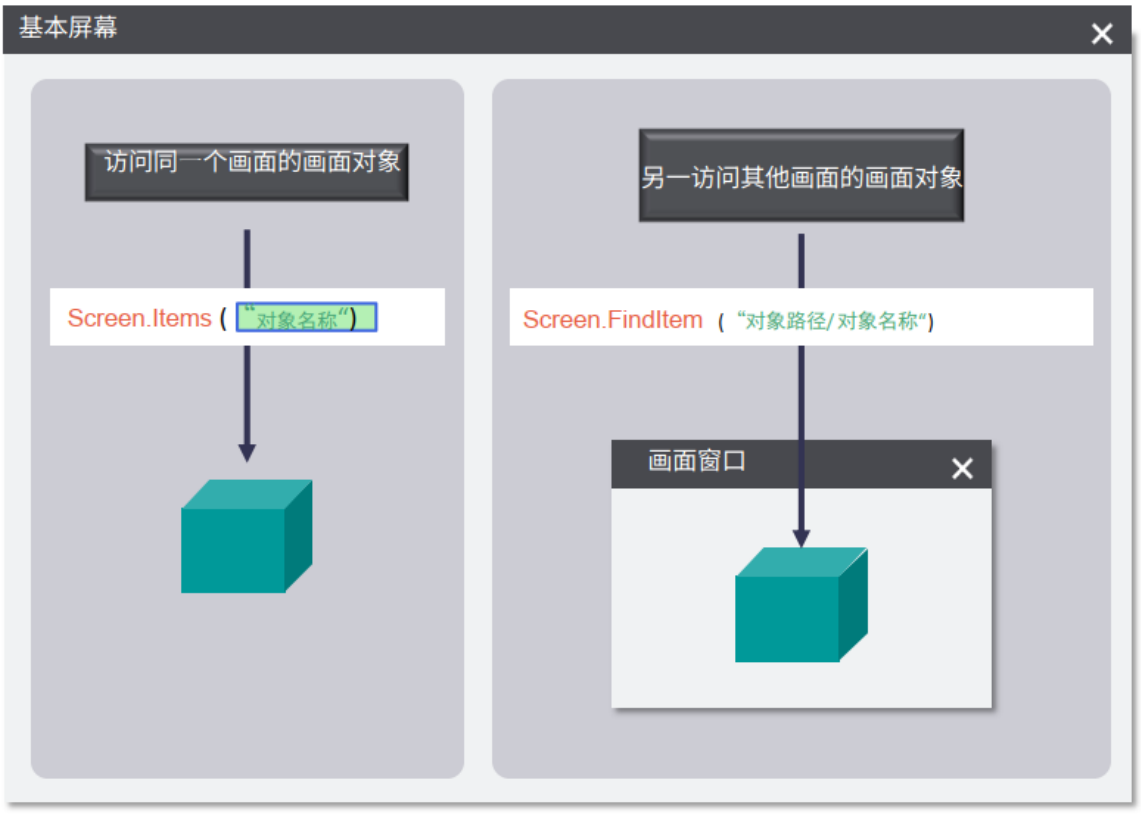


图 3-86 Screen.Items 与 Screen.FindItem 的用法

3.5. 其他

- 减少文本列表中的空条目
- 注意系统限制

画面

	7-12" Unified 精智面板	15-22" Unified 精智面板
工程组态系统中的最大大小	20,000 * 20,000 像素	
运行系统中的最大大小	20,000 * 20,000 像素	
画面数目	1200	
下级画面窗口数	10	
每个画面的对象数目	800	1200
每个画面 "控制" 区域的对象数目	40	80
每个画面的变量数目	600	800

图3-95: 在\9\ 中定义的系统限制摘录

- 仅适用于PC-RT：在生产中运行的项目应禁用脚本调试器

3.5.1. 采集周期

注册 PLC 变量后，将通过HMI 配置该值从 PLC 更新到HMI 的频率。这个时间被定义为采集周期。如果 PLC 在一个周期后检测到数值变化，则将数值从 PLC 传送到 HMI。 如果数值保持不变，则 HMI 和 PLC 之间不进行通信。

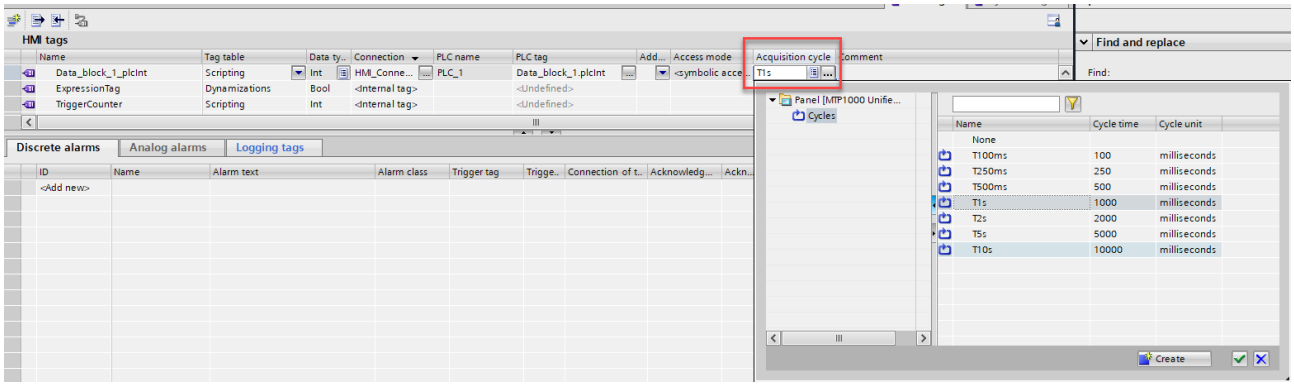


图 3-96 连接 PLC 的 HMI 变量的采集周期

一般来说，建议配置较高的采集周期，因为这样可以减少通信负荷。

采集周期

一般情况下，PLC 变量的采集周期较长/较高，并使用点动系统功能，以防止覆盖值出错。

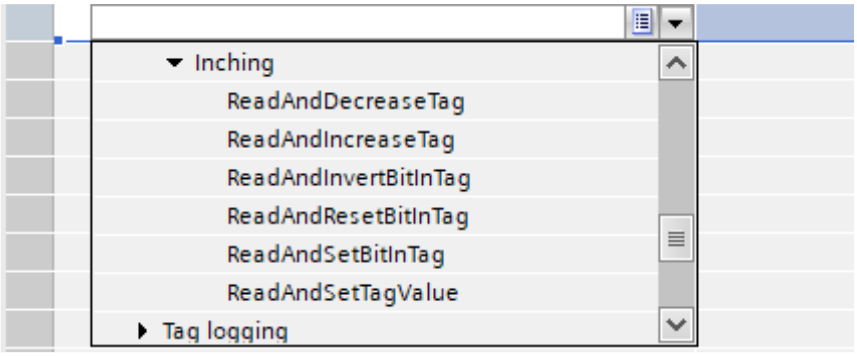


图 3-97 PLC 变量的点动系统功能

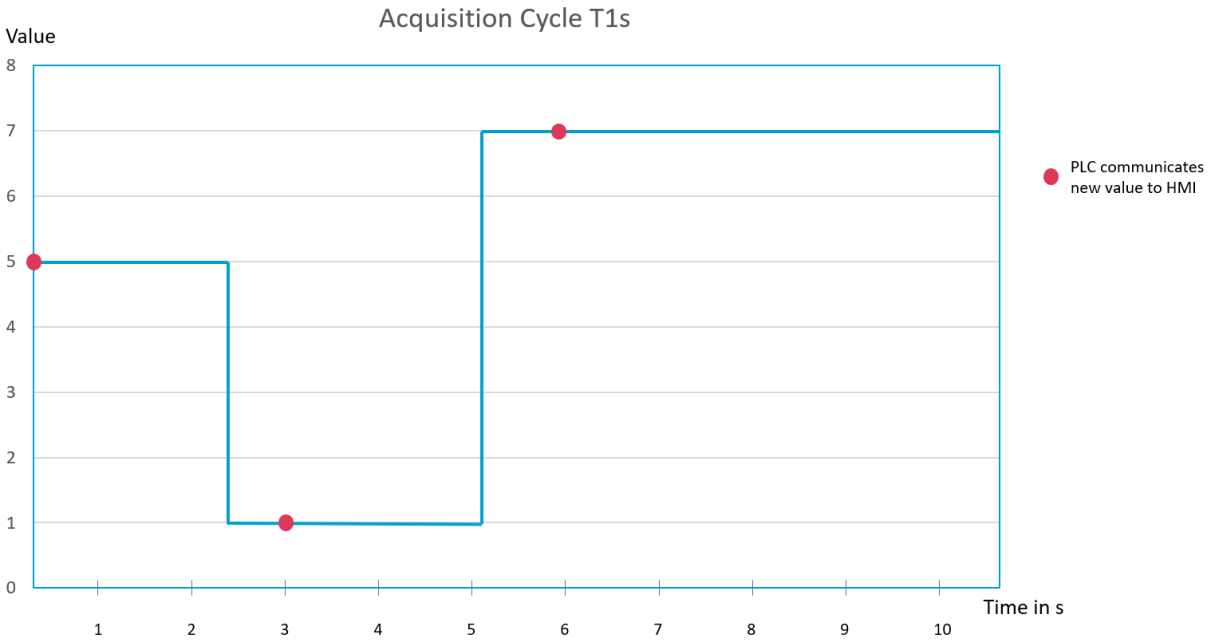


图 3-98 采集周期为 T1s 的变量的 HMI-PLC 通信

请注意，具有较高采集周期的PLC变量更新频率较低，因此值的变化仅在当前周期时间结束后的运行时可见

3.5.1.1. 使用案例：在画面加载事件中写入 PLC 变量

定义

画面窗口内的画面也可通过编号设置。如果根据变量，画面窗口内应显示不同的画面，则有必要在画面加载事件中读取和/或写入 PLC 变量。

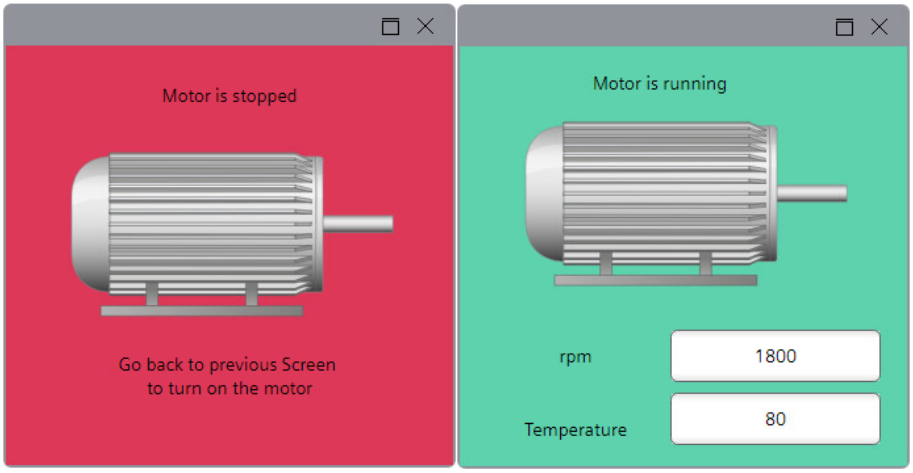


图 3-99 应显示的不同画面

解决方案

在接下来的画面加载事件中，会写入一个 PLC 变量，用于动态化画面。

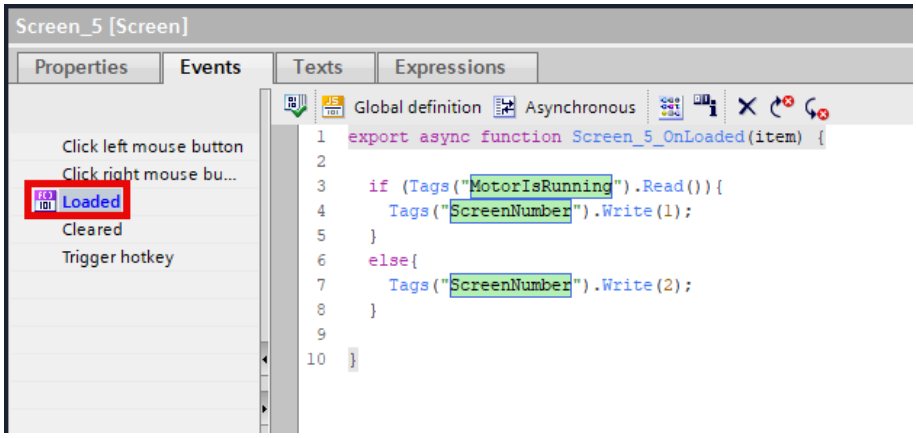


图 3-100 在画面加载事件中写入 PLC 变量

此画面编号用于动态化画面窗口内显示的画面。

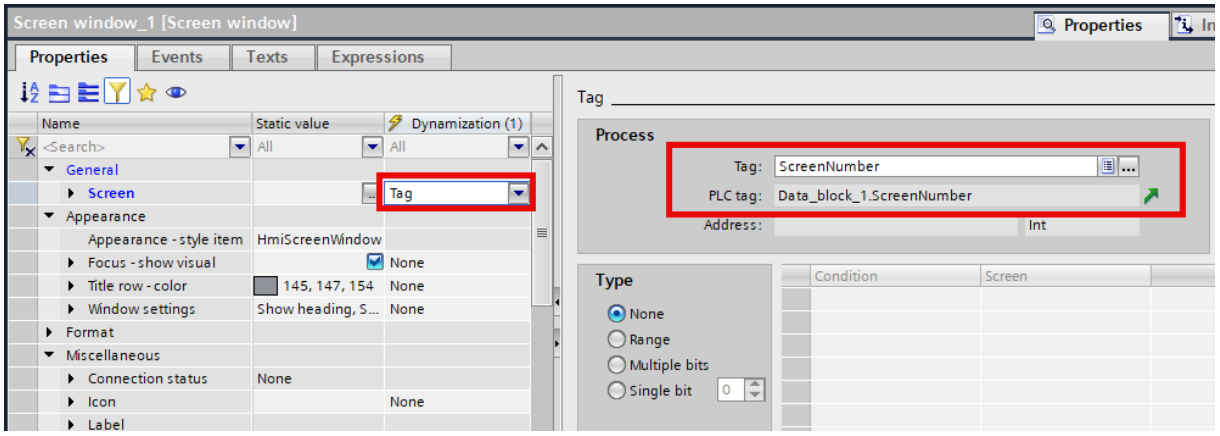


图 3-101 通过画面编号动态化画面

由于写入的变量 “ScreenNumber ”是 PLC 变量，新值将在一个完整的采集周期后显示。因此，请为在画面加载事件中写入的 PLC 变量配置一个较小的采集周期。

采集周期加载事件

为需要在画面加载事件中写入的 PLC 变量配置小/短采集周期加载事件。

3.5.2. 使用案例：带多路复用功能的 PLC UDT 数组

定义

要使用变量值描述对象（如电机），UDT 是对象数据结构化的好方法

Data_block_1

	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Static							
2	Motor1	"UDTMotor"						
3	Name	WString	WSTRING#''					
4	Speed	Real	0.0					
5	Acc	Real	0.0					
6	Motor2	"UDTMotor"						
7	Name	WString	WSTRING#''					
8	Speed	Real	0.0					
9	Acc	Real	0.0					

图3-102 小型电机PLC UDT

如果该对象有很多实例（例如20 个），但Runtime 不需要同时提供所有实例的数据，则可以选择多路复用。

这样，只需处理该数组中的一个子集（如5 个电机），当需要另一个电机的数据时，只需通过选择数组中的另一个索引来更改对该实例的引用。

数据块_1

	名称	数据类型	起始值	保持	从 HMI/OPC...	从 H...	在 HMI ...	设定值	监控
1	Static								
2	Motors	Array[0..20] of "Mo...							
3	Motors[0]	"Motor"							
4	Name	WString	WSTRING#''						
5	Speed	Real	0.0						
6	Acc	Real	0.0						
7	Motors[1]	"Motor"							
8	Motors[2]	"Motor"							
9	Motors[3]	"Motor"							
10	Motors[4]	"Motor"							
11	Motors[5]	"Motor"							
12	Motors[6]	"Motor"							
13	Motors[7]	"Motor"							
14	Motors[8]	"Motor"							
15	Motors[9]	"Motor"							
16	Motors[10]	"Motor"							
17	Motors[11]	"Motor"							

图3-103: 20 个电机的PLC UDT 数组

有时，UDT要复杂得多，并且存储了大量数据，这些数据可能是PLC所必需的，但不是HMI所必需的。

图3-104: 复杂的PLC UDT包含的数据多于HMI 所需要的

解决方案

如果可视化中只需要UDT 实例中的少量数据，则不要使用多路复用。即使UDT 中只有一个数据点与 HMI 属性相连，在更改UDT 的具体实例时，也会从PLC 侧加载整个结构，而不仅仅是所需的部分。

对于存储在复杂PLC UDT 中的数据，应将每个UDT 作为一个独立实例加载，而不是使用具有多路复用功能的数组。每次更改索引时，PLC 中的整个结构都会被刷新，即使该结构未被使用。

PLC UDT 的多路复用
只有在简单的PLC UDT 或在可视化中需要所有UDT 实例化的数据时，才使用多路复用功能。

3.5.3. 使用案例：多语言

定义
当项目中有多种语言时，不同的语言会适合使用不同的字体，来实现特定字符的显示完整性。因此，当添加一种具有不同字体的语言时，多个字体也会被添加到项目中，并下载到设备上，导致占用一些运行资源。

解决方案
为了节省运行时的内存，在运行系统设置中只激活当前项目需要的语言。

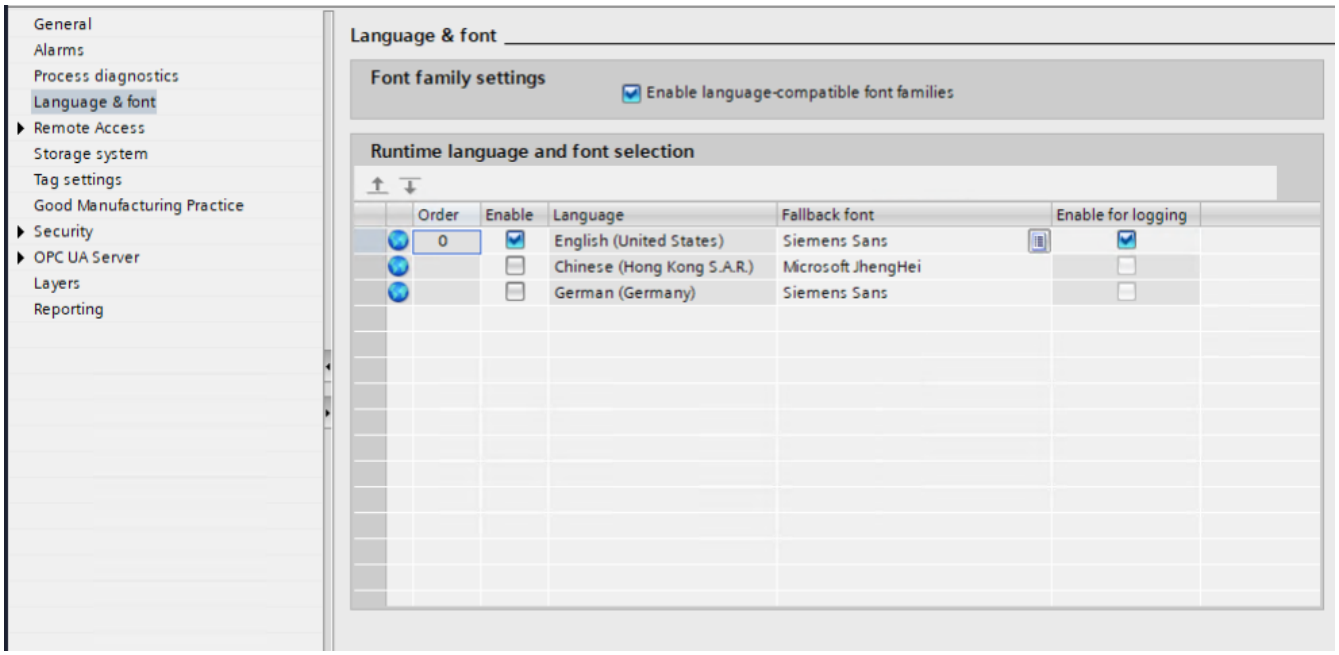


图 3-97 HMI 设备的语言选择

语言和字体

如果项目中某种语言已经停用，那么在运行系统设置中需要删除对应的字体，除了 Siemens Sans!

4. 现有项目分析

在已有项目中，无论运行过程是否出现问题，对项目在工程组态中是否存在改进机会进行分析总是有益的。本节将提供一份指南，向您展示所有进一步分析 WinCC Unified 项目的可能性，给出了一个可行的逐步分析过程的概述。

如果Runtime 出现已知问题（画面加载缓慢、某些功能无法正常工作），您可以直接跳转到对象或脚本分析。除此以外，您可以在目标设备上运行该项目进行测试，检查是否一切正常。

对于分析而言，熟悉项目Runtime 的一般工作流程（如运行脚本的顺序）和后台运行的内容（如定时器的循环脚本）也很重要，当发现问题时能找到工程组态中的相关位置（脚本、组态、任务）。如果对Runtime 的脚本和过程处理顺序了解较少，使用谷歌Chrome的脚本调试器也是一个很好的选择，能在画面加载期间通过单击按钮或特定画面变化事件自动定位所有已执行的脚本，

另一种方法是将跟踪放入脚本中，并在触发时通过RTIL Trace Viewer 进行检查。

一旦熟悉了项目中的基本脚本执行和流程，就为进一步分析奠定了良好的知识基础。

下面的方案显示了一个可行的过程，即按照什么顺序以及如何选择工具来分析项目，何时查看运行，何时更关注项目的脚本部分。分析也包括两个部分，即对象分析和脚本分析。可以根据项目的具体情况改变执行顺序，即先分析哪个部分。

流程图中提到的所有过程将在后续章节中详细介绍。

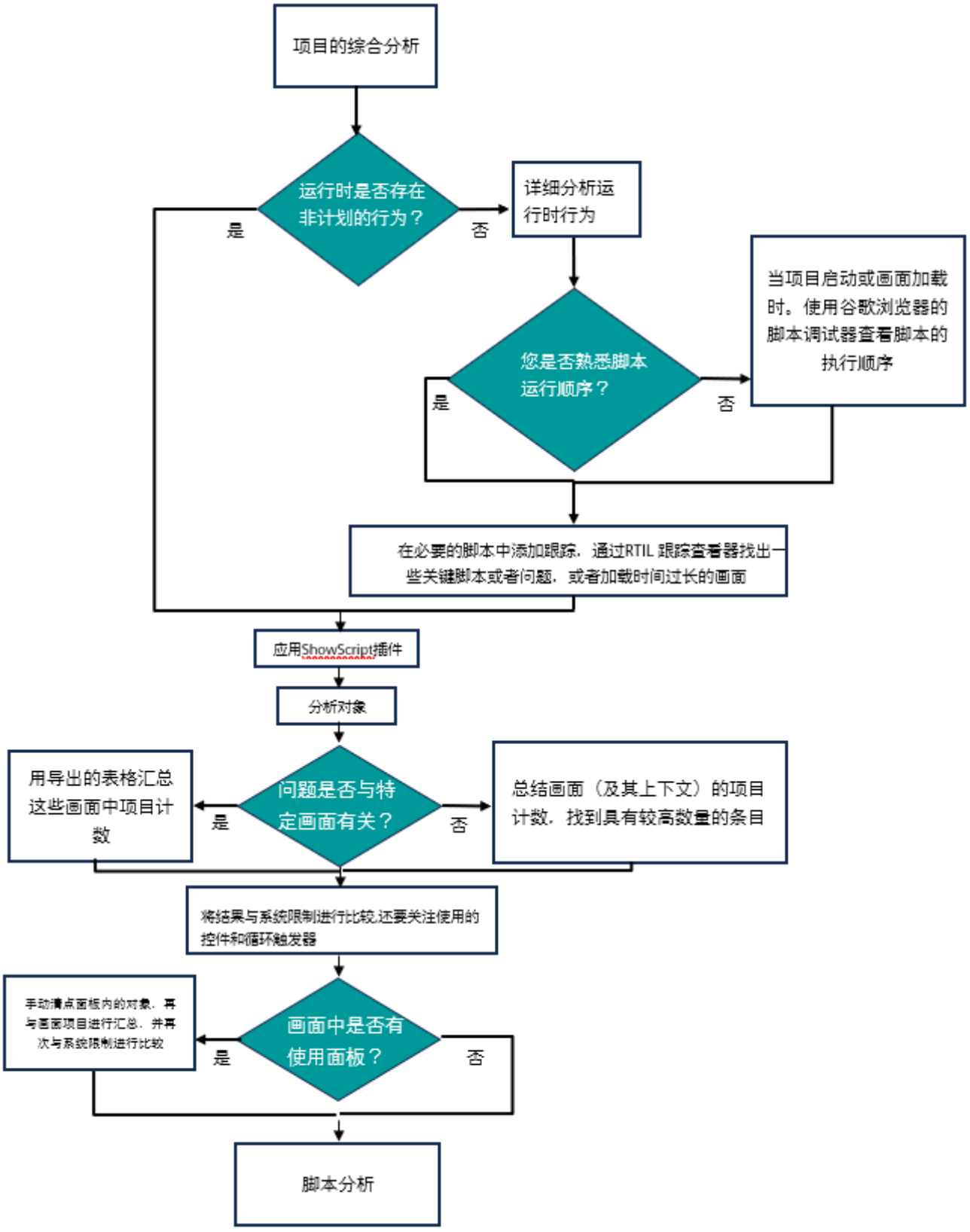


图4-1 项目分析启动流程图

下面的流程图显示了脚本分析的各种可能性。

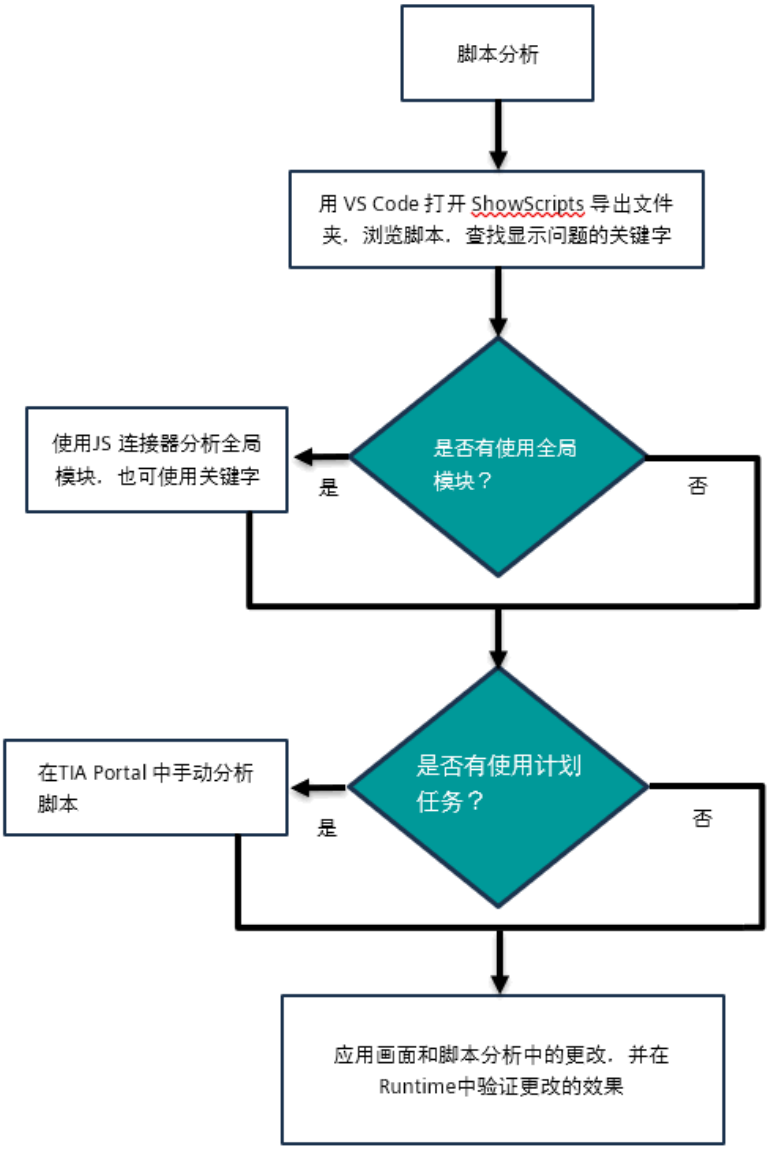


图4-2 项目分析的脚本分析流程图

4.1. SmartAdvisor 插件

SmartAdvisor Add-in 是一个模块化工具，它使用“TIA Openness”接口来支持 TIA Portal 在项目分析以及 WinCC Unified 设备项目功能上的优化。该应用程序可分步执行，并为已执行的流程提供详尽反馈

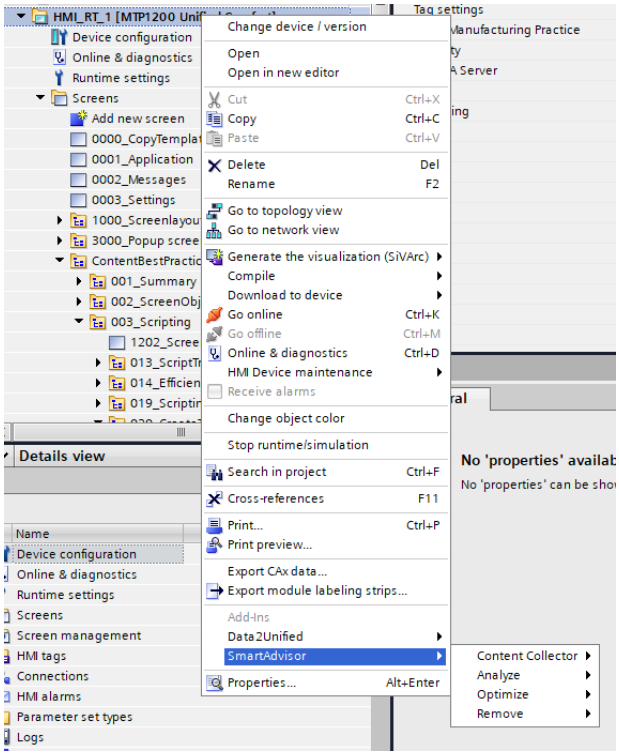


图 4-3 Smard Advisor 插件中的 4 个子菜单

综上所述，可提供以下分析与优化功能：

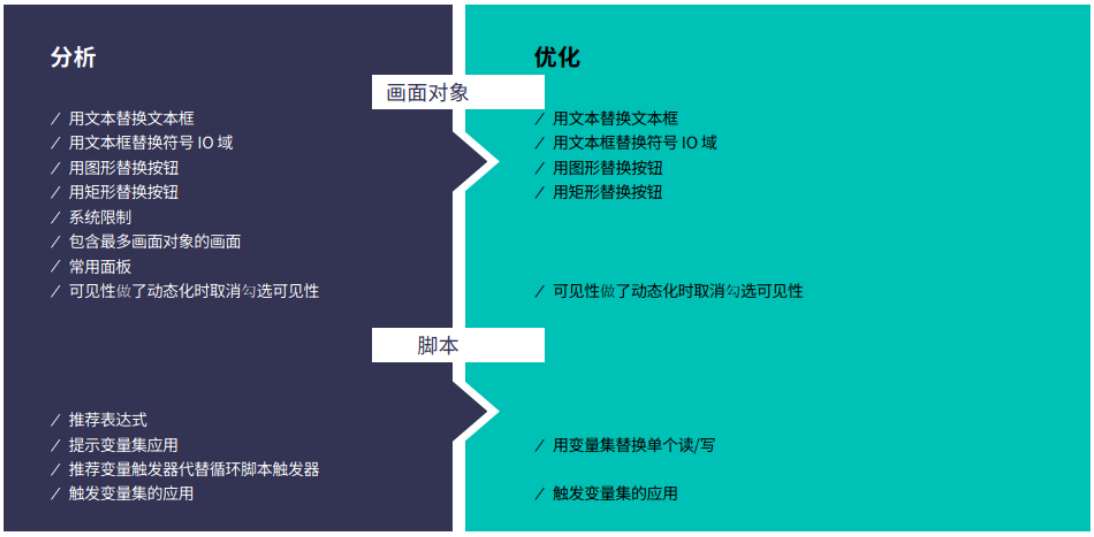


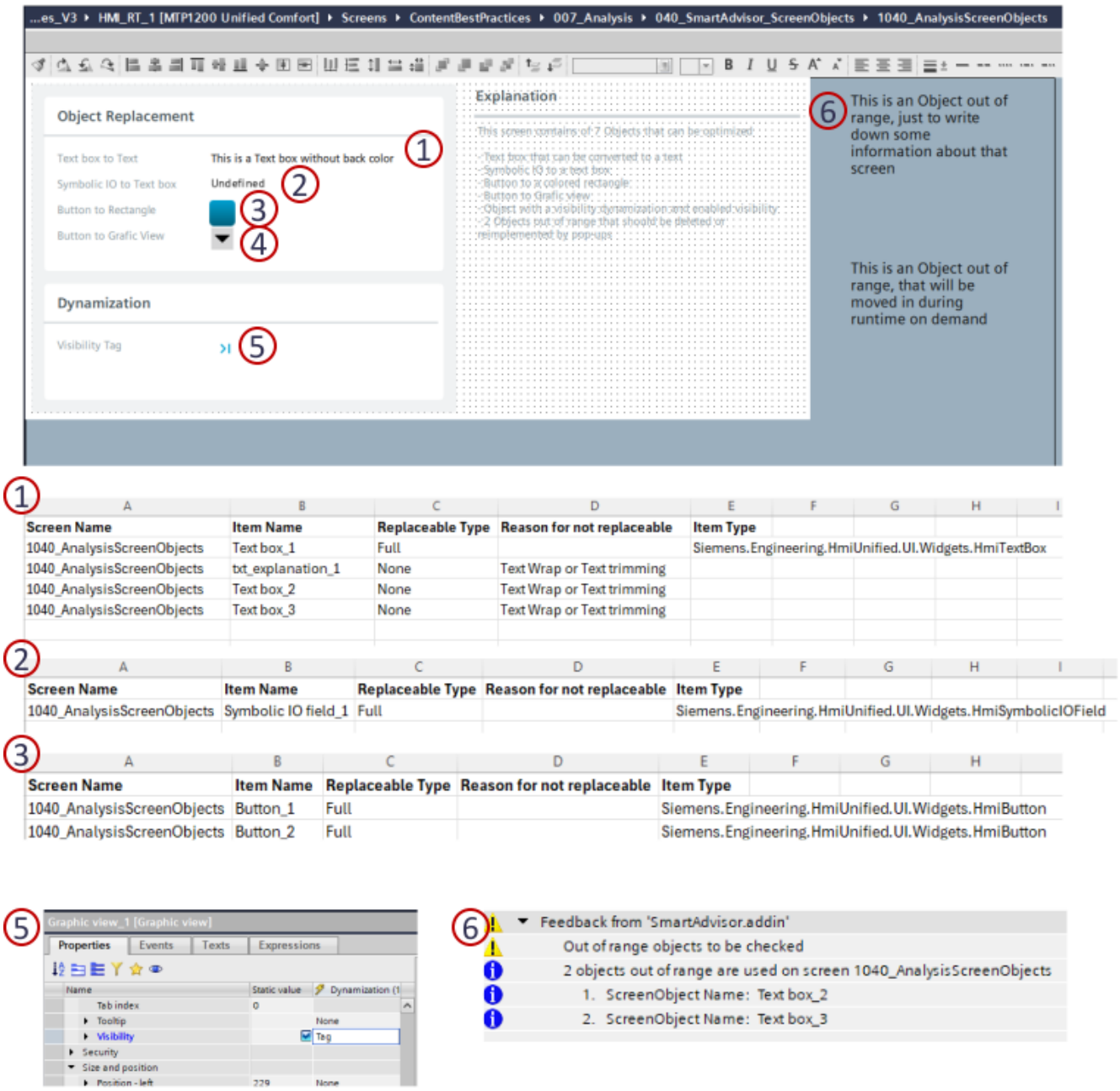
图 4-4 SmartAdvisor 分析和优化功能

注意

所有关于下载、安装和应用的详细信息都可以在

SmartAdvisor 手册 > 链接

4.1.1. SmartAdvisor 画面对象优化



1

Screen Name	Item Name	Replaceable Type	Reason for not replaceable	Item Type
1040_AnalysisScreenObjects	Text box_1	Full		Siemens.Engineering.HmiUnified.UI.Widgets.HmiTextBox
1040_AnalysisScreenObjects	txt_explanation_1	None	Text Wrap or Text trimming	
1040_AnalysisScreenObjects	Text box_2	None	Text Wrap or Text trimming	
1040_AnalysisScreenObjects	Text box_3	None	Text Wrap or Text trimming	

2

Screen Name	Item Name	Replaceable Type	Reason for not replaceable	Item Type
1040_AnalysisScreenObjects	Symbolic IO field_1	Full		Siemens.Engineering.HmiUnified.UI.Widgets.HmiSymbolicIOField

3

Screen Name	Item Name	Replaceable Type	Reason for not replaceable	Item Type
1040_AnalysisScreenObjects	Button_1	Full		Siemens.Engineering.HmiUnified.UI.Widgets.HmiButton
1040_AnalysisScreenObjects	Button_2	Full		Siemens.Engineering.HmiUnified.UI.Widgets.HmiButton

5

6

图 4-5 画面对象优化演示

当使用 Demo project 时，画面对象分析结束后，会给出以下示例：

1. 将没有背景颜色的文本框替换为文本
2. 将仅作为文本输出的符号 IO 域替换为文本框
3. 将只有背景颜色的按钮替换为矩形
4. 将带有图形的按钮替换为图形视图
5. 当可见性做了动态化时取消勾选可见性
6. 指示超出范围的对象

注意 WinCC Unified 的演示项目可以在这里找到： > 链接

4.1.2. SmartAdvisor 脚本优化

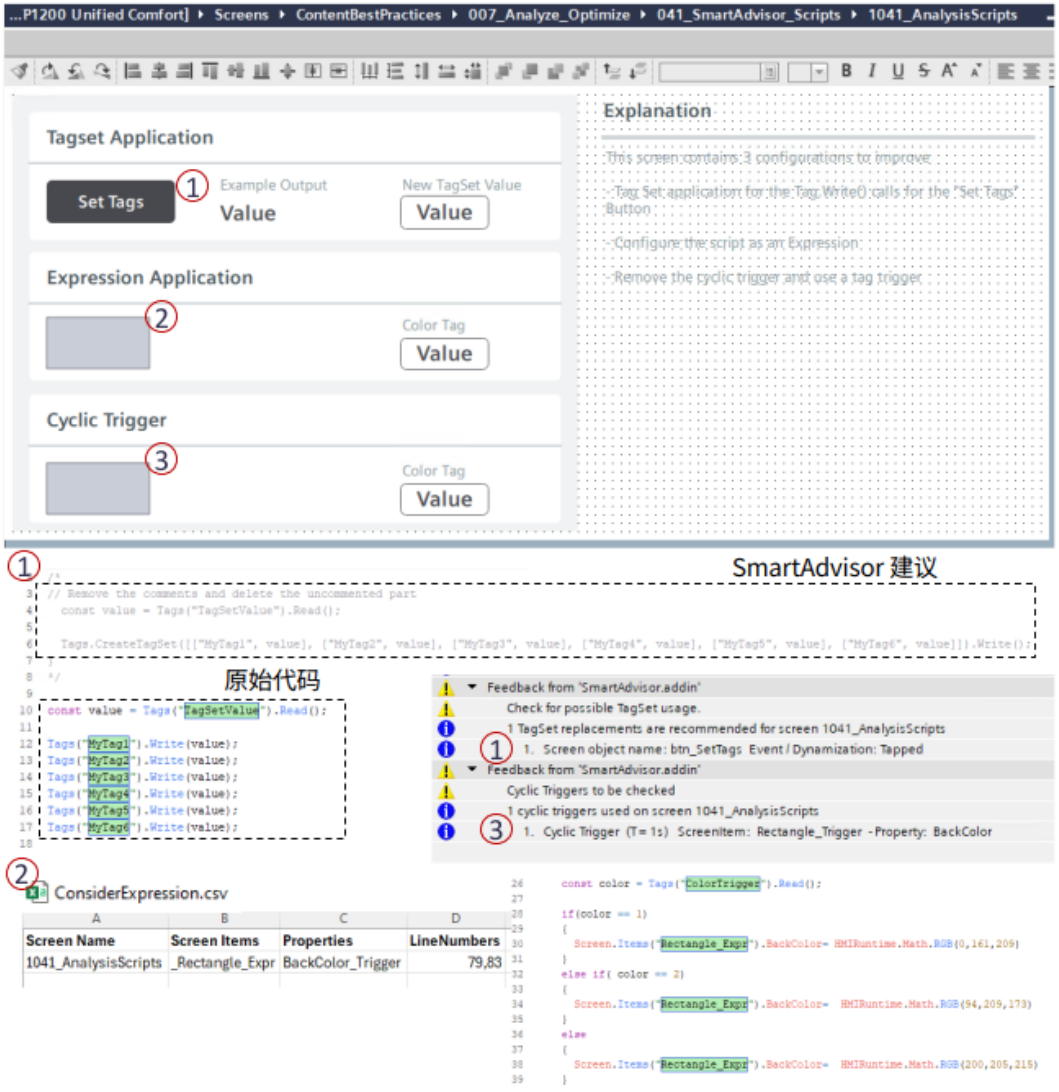


图 4-6 脚本优化演示项目

当使用 Demoproject 时，脚本分析结束后会给出以下示例：

- 使用 tagset()应用代替单个变量的读/写调用
- 属性动态化中建议使用表达式代替 if else 语句
- 用变量触发器替换的周期循环触发器。

4.2. WinCC Unified JS CONNECTOR

画面相关脚本的导出和分析的第一步是使用 ShowScripts 插件。如果在画面相关脚本中检测到使用了全局模块脚本，则可以使用"Simatic WinCC Unified JS Connector"，这是一个 Visual Studio Code 扩展程序，可用于导出、编辑和导入这些全局模块脚本。

注意 自WinCC Unified ES V19 起支持WinCC Unified JS Connector 扩展使用。

下面是使用全局模块的示例。如果要查看导入模块中使用的脚本内容， 可以使用 WinCC Unified JS Connector。

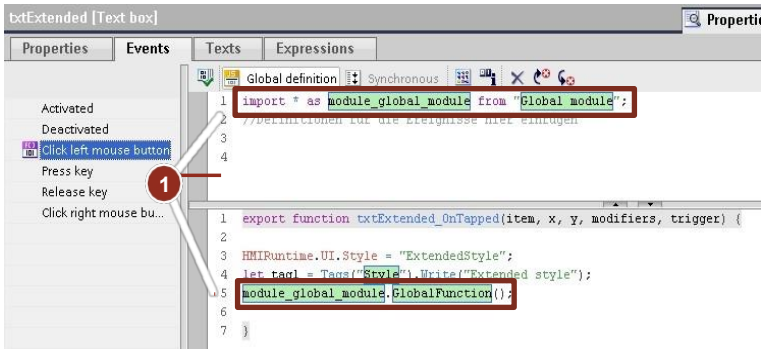


图4-29 WinCC Unified 脚本编辑器中全局函数的使用

WinCC Unified JS CONNECTOR 是免费的，可在西门子工业在线支持（SIOS）的单独条目中获取。通过该扩展应用，用户可以在工作过程中对项目进行分析和优化：

1. 从TIA Portal项目导出全局模块中的脚本

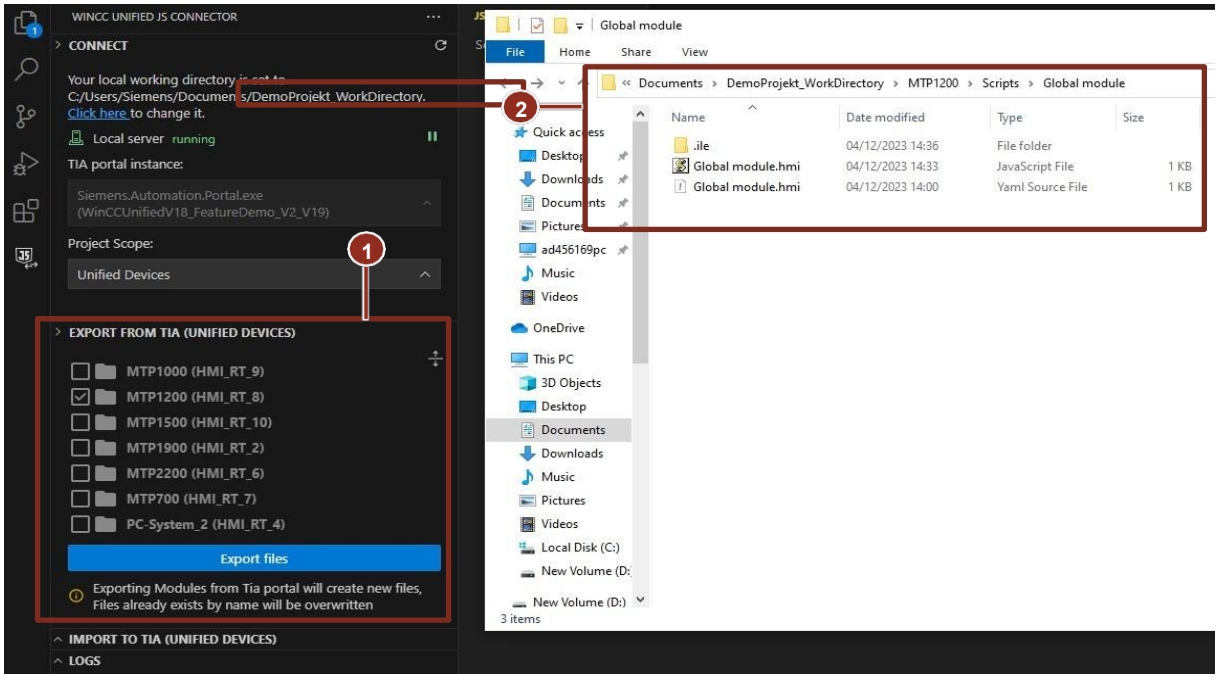


图4-30 扩展中的导出的功能(1) 在工作目录(2) 下的导出文件

2. 分析和调整 VS Code中的脚本（按照第 3 章中的最佳方法）

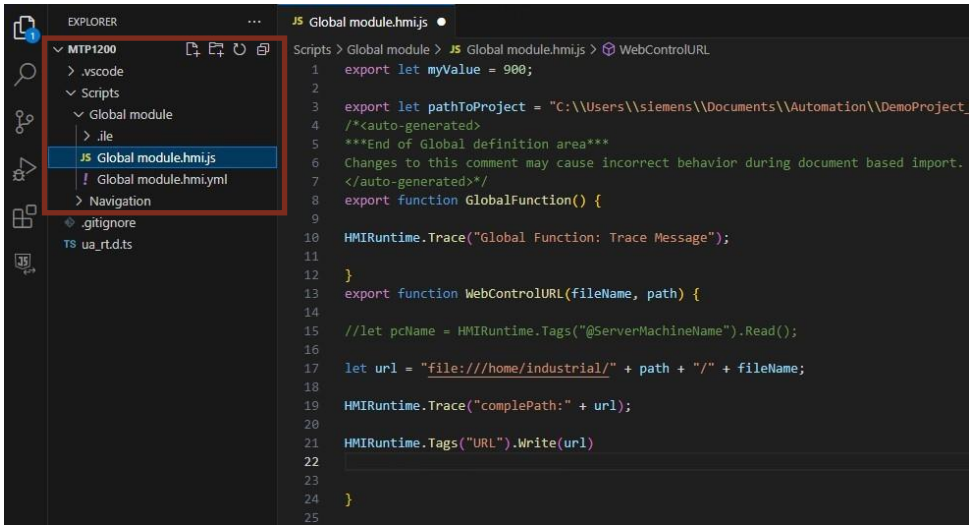


图4-31 通过VS Code分析导出的全局模块脚本

3. 将经过调整的脚本导入博途项目

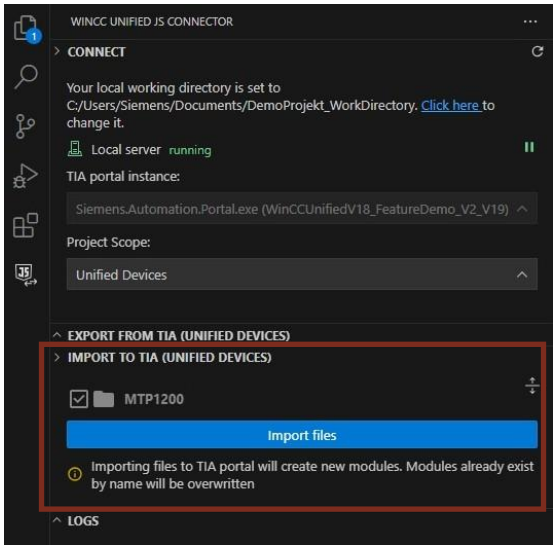


图4-32 将脚本导入 TIA 博途项目

有关 WinCC Unified JS CONNECTOR 安装和使用的详细信息，请参阅SIOS 条目中提供的相关手册。

4.3. 运行分析

如果您已经使用上述的ShowScripts 插件和WinCC Unified JS CONNECTOR 成功分析了项目，并在画面工程或脚本代码中应用了更改和优化，但在画面变化时仍有可能出现错误行为，或者没有充足时间加载画面。

为了进一步挖掘项目优化的潜力，可以使用以下工具更详细地分析有问题的画面更改：

1. RTIL Trace Viewer （适用于所有 Unified 设备）
2. WinCC Unified 的脚本调试器（仅适用于 Unified PC 项目运行和工程PC 上的面板仿真）

4.3.1. RTIL Trace Viewer

RTIL Trace Viewer 是WinCC Unified 安装程序中的一个外部应用程序。它可显示Unified Basic/Comfort Panel 或 PC Runtime在Runtime 的所有可用跟踪消息。

注意

在WinCC Unified 设备上使用RTIL 跟踪查看器的设置步骤，在下面的常见问题解答中有详细说明：

与 WinCC Unified精智面板或WinCC Unified PC Runtime 一起使用跟踪查看器

Unified精智面板的操作步骤同样适用于Unified精简面板。

使用 RTIL Trace Viewer 进行项目分析

Trace Viewer 可用于进一步分析以下情况：

1. 在捕捉画面变化时，可能会出现更多与脚本执行或项目配置相关的错误或警告。这些跟踪信息展示了对整体画面性能产生影响的因素，需要尽快解决
2. 这些跟踪信息会对整体画面更改性能产生一定影响，需要尽可能加以解决。

下面是此类错误的一个示例，其中一个已配置的脚本试图访问对象 （"Button_4"）。项目 Runtime 此对象不存在，因此会显示错误信息。

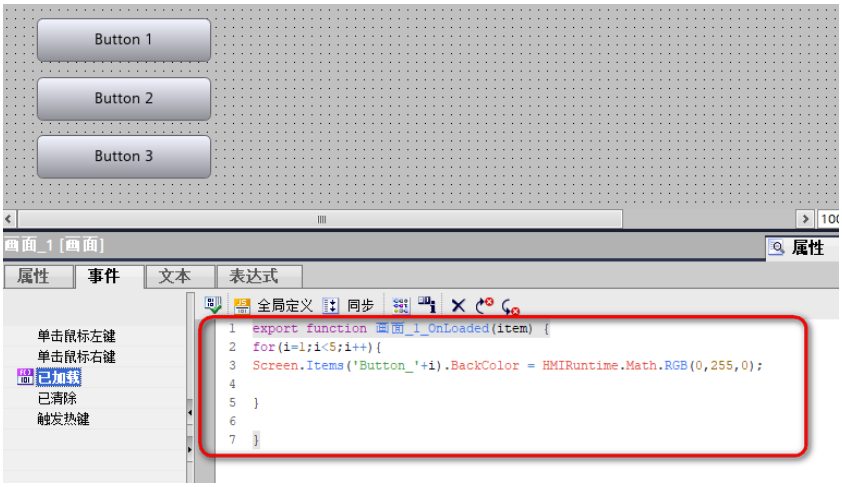


图4-33 脚本访问画面上的按钮

3. 显示在RTIL Trace Viewer 中的脚本执行结果是便于理解的，只需要通过调整脚本循环来解决错误。

图 4-35 脚本的执行结果在 Trace Viewer 中应用

注意

为了更好地分析 RTIL 跟踪查看器中的跟踪，可以在默认视图中应用特定的显示过滤器。为了查看只与脚本相关的跟踪，可以将"子系统"过滤器设置为"ScriptFW"。

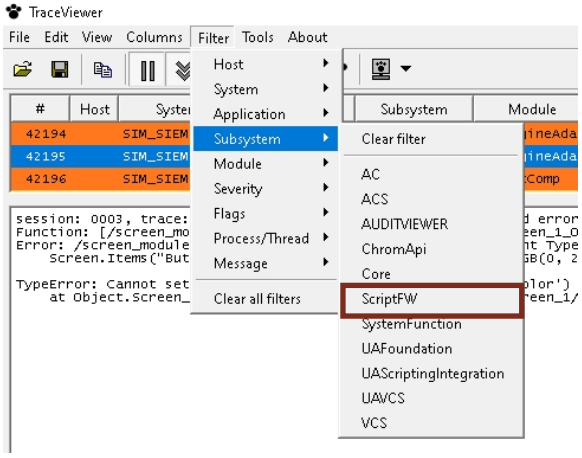


图4-35 RTIL Trace Viewer 中的脚本过滤器

4. Trace Viewer 是检测和了解项目中脚本可能出现的故障的方法。由于Unified Basic/Comfort Panel 硬件无法进行进一步调试，因此将跟踪信息与RTIL Trace Viewer 结合使用有助于实现以下目的：
    - 查看在特定操作场景下哪些脚本会执行
    - 检查这些脚本的执行频率
    - 脚本的处理时间

对于 Unified PC Runtime，下一章将介绍脚本调试器，它可以替代 Trace Viewer 。

注意

有关跟踪查看器的使用和跟踪信息配置的更多信息，请参阅 Tricks for Scripting (JavaScript) , 第5.16.2章节"使用 RTIL 跟踪查看器进行诊断"。

4.3.1.1. 附加标志

为进一步分析，可在跟踪查看器中启用其他选项，以获取以下信息：

- 查看哪些脚本会在某些运行场景中被执行
- 查看这些脚本的执行频率
- 脚本的处理时间
- 画面构建过程中读取的变量数量

性能标志

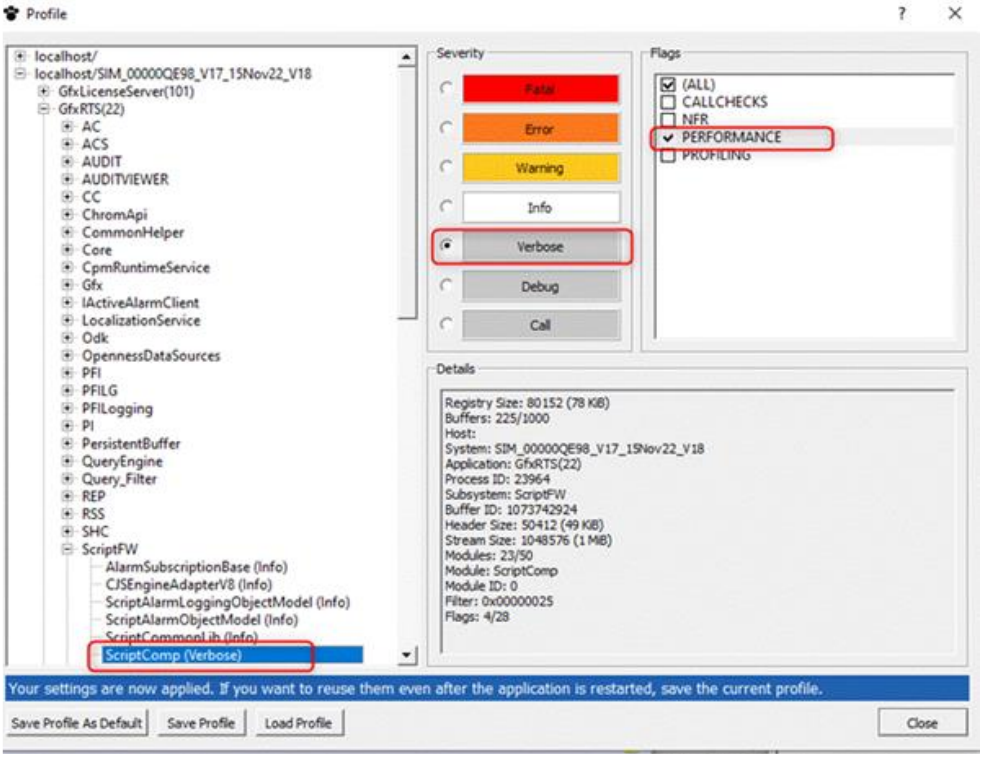


图 4-36 设置性能标志

激活该项后，您就可以全面了解正在执行的脚本数量。这样，您就可以推断脚本是否重复执行，从而判断是循环触发还是周期触发。此外，您还会收到以下信息，如 DoCommand（脚本持续时间）和 ExecuteCommand（脚本触发持续时间）时间。您可以分析这些时间，并通过脚本优化来改进它们。

STAT_HISTO 标志

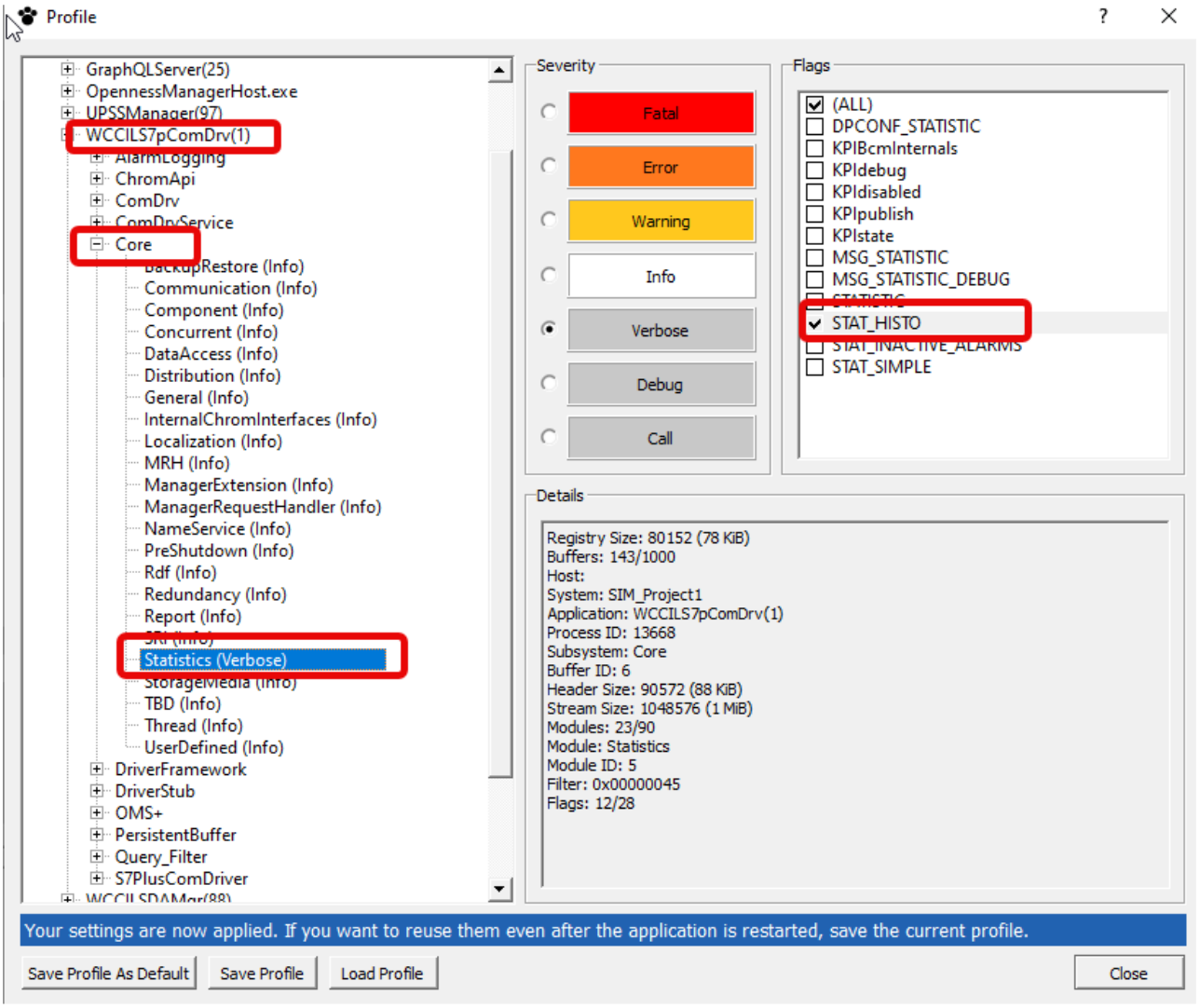


图 4-37 启用 STAT_HISTO 标志

使用此选项您可以大致了解在画面构建（画面加载）过程中读取了多少变量。画面构建（画面加载）过程中读取的变量数量会影响性能。

4.3.2. 脚本调试器

在工程计算机中运行Unified PC和面板仿真，可以结合使用Google Chrome 浏览器的脚本调试器。

调试器可为脚本分析提供更多可能性，并提供设置断点和逐步执行等典型功能。

注意 有关 WinCC Unified 中脚本调试器的设置和使用的详细说明，请参阅WinCC UnifiedWinCC Unified V19 手册册

使用脚本调试器分析画面变化

通过 Unified PC Runtime 的脚本调试器，可以进一步评估具体操作情况：

- 在特定操作场景下查看执行的脚本。例如，在已加载事件中设置了断点，通过点击单个步骤，也会显示脚本所有的后续执行情况。
- 通过计算脚本中达到断点的频率，检查这些脚本的执行频率。

如果要调试画面变化时执行的脚本，则需要以特定方式准备项目，以便能同时调试执行的脚本。下面的步骤展示了如何在基本画面的加载事件中配置一个断点，并再次触发加载，逐步查看初始画面加载过程中执行的所有脚本，因为这是上述两种配置中难度较大的一种。

1. 在运行管理器中激活脚本调试器

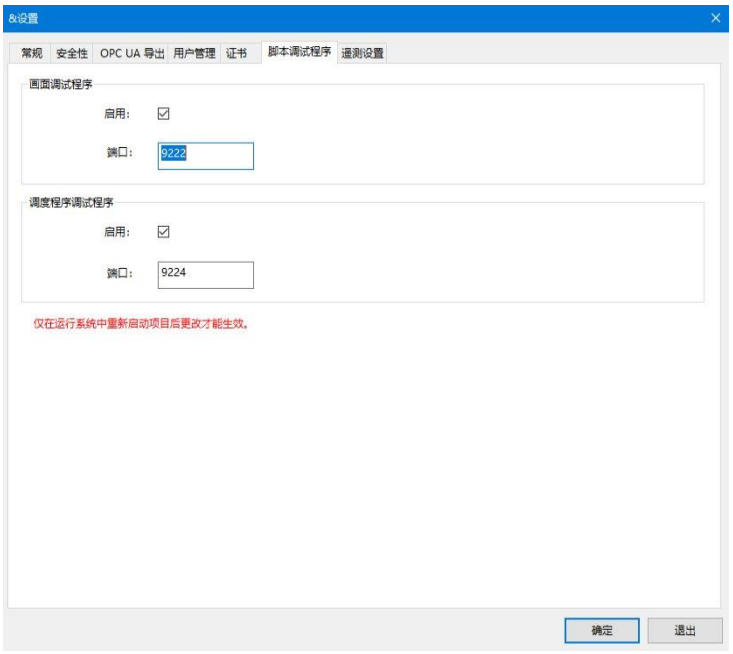


图4-38 在运行管理器的设置中启用脚本调试器

2. 在项目基础画面的已加载事件中配置代码。这将用于在基础画面开始加载的位置设置断点。定义一个在项目 Runtime 可触发的事件（例如，通过鼠标点击画面对象的事件），该事件将基础画面从一个非当前的基础画面切换到原始基础画面。如果只是将基础画面再次设置为原始画面，则不会触发已加载事件。

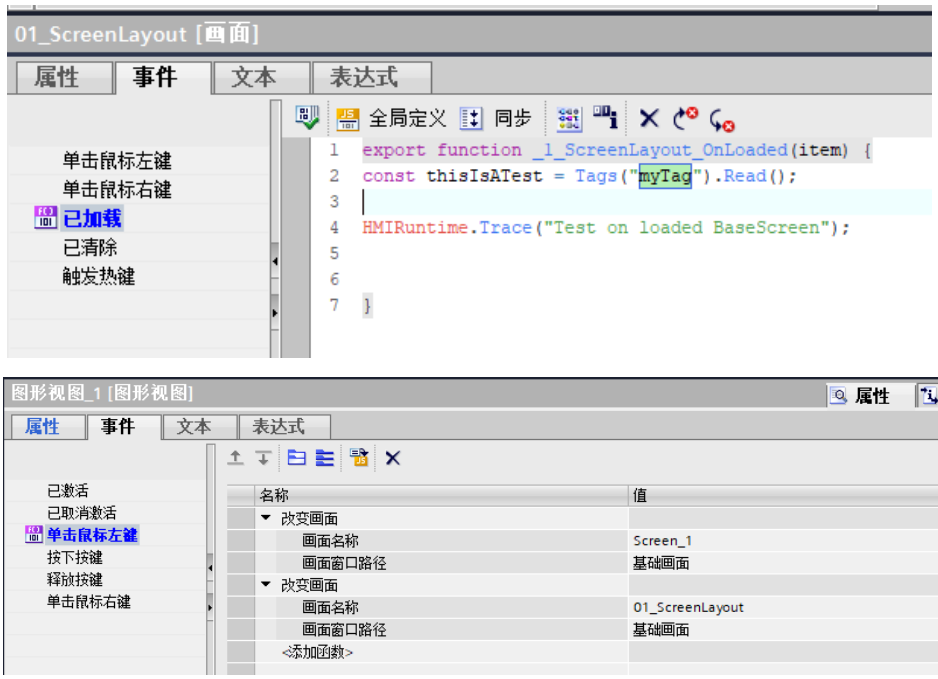


图 4-39 刷新基础画面和已加载事件的代码

3. 启动运行

4. 打开另一个浏览器窗口并输入：chrome://inspect/

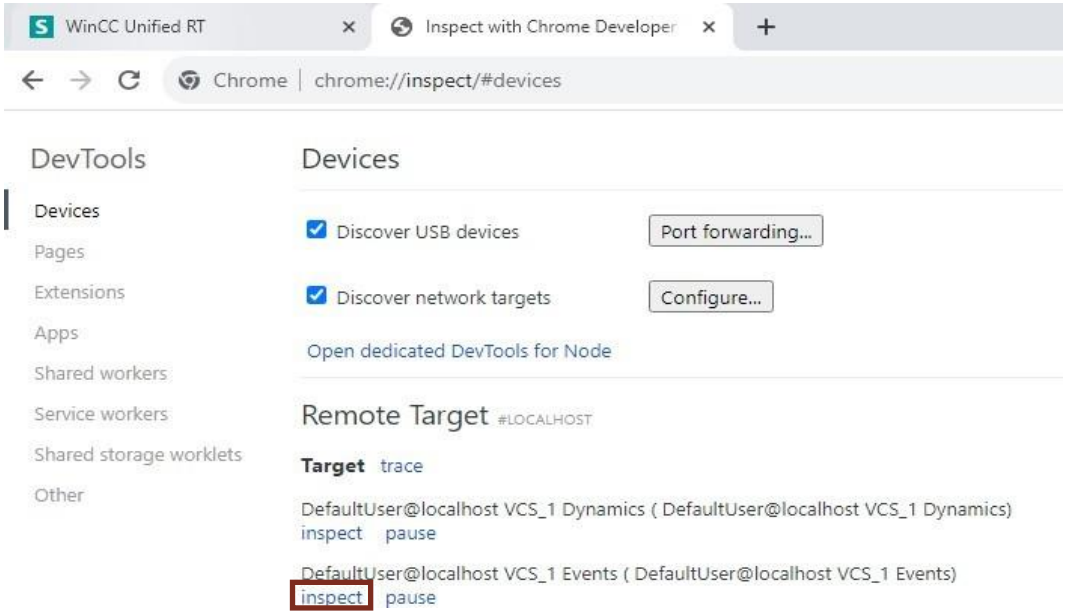


图4-40 Chrome 浏览器开发工具页面

5. 点击事件上下文的检查链接。这样，一个新的浏览器客户端就启动了

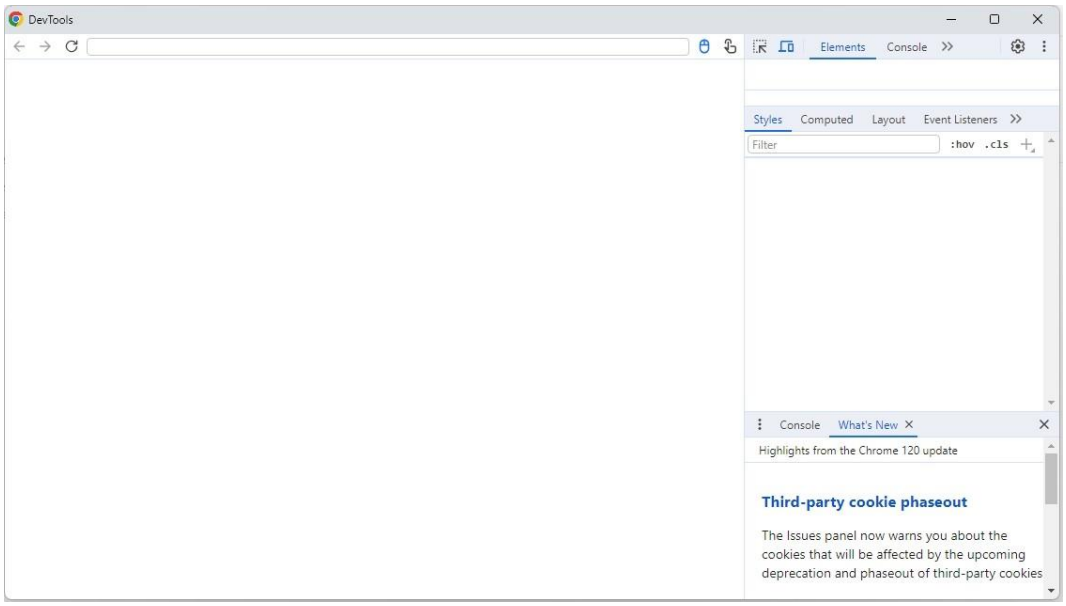


图4-41 Dev Tools 浏览器窗口

6. 将右侧窗口向左拖动，配置浏览器窗口，然后切换到源代码选项卡。展开（no domain）目录并选择基础画面的Event.js。在基础画面中出现的是之前定义好的已加载事件中的代码。此列表显示的是当前已加载的所有脚本。因此，列表会根据当前打开的画面而有所不同。

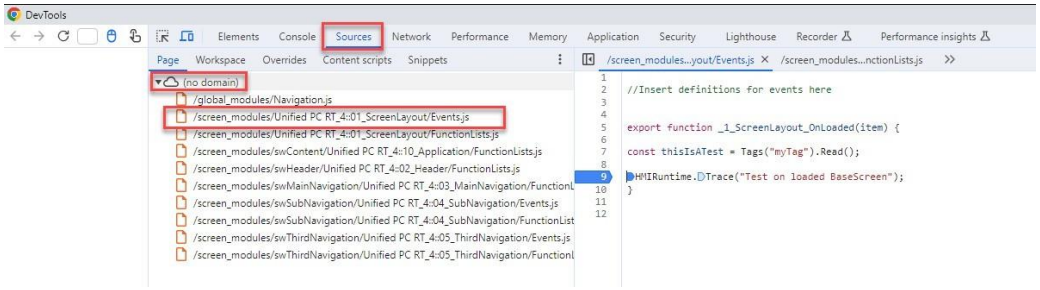


图4-42 脚本调试器- 设置断点

7. 在已加载事件中设置中断点
8. 返回Runtime， 触发基础画面更改事件
9. 调试器浏览器窗口将自动打开并跳转到断点。你还可以在右侧看到已执行的值。点击"Step into next function（进入下一个功能）"按钮，即可逐步浏览脚本。

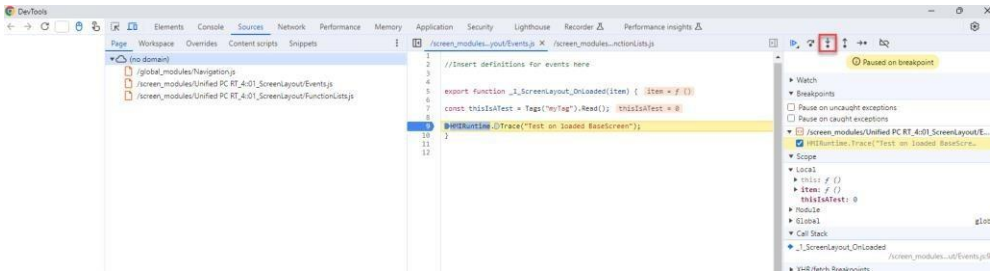


图4-43：脚本调试器- 跳转到断点

您可以跟踪全局模块的加载情况

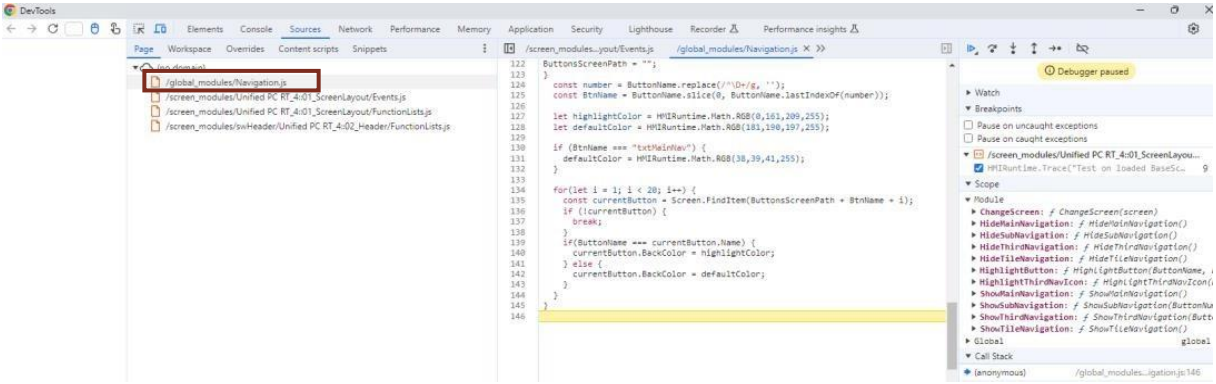


图4-44 脚本调试器- 加载脚本模块

嵌套画面窗口和面板的已加载事件

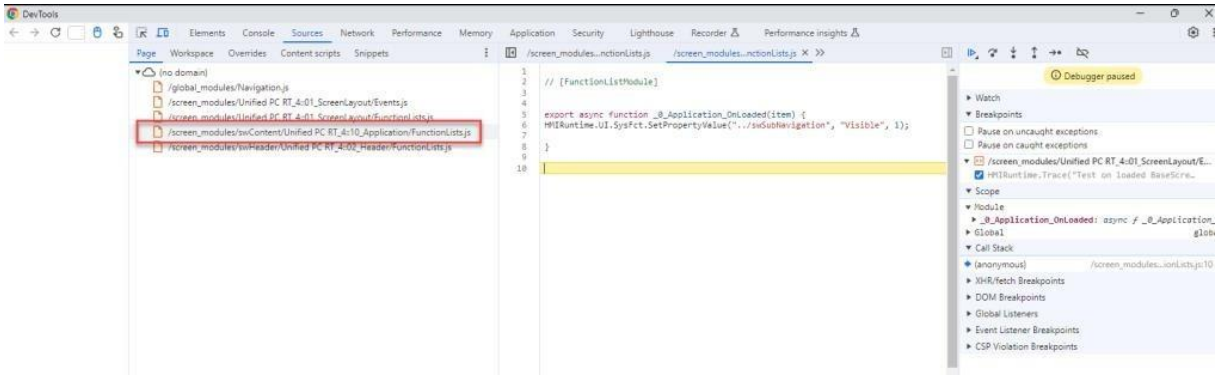


图4-45 脚本调试器--已加载事件画面窗口

执行的每个脚本都会显示出来。一旦您想结束调试并立即执行其余脚本，请单击"继续执行脚本"按钮。如果定义了多个的断点，调试器将跳转到下一个断点。

图4-46：继续执行脚本

10. 每次新的下载都需要打开一个新的 Dev Tools 会话。这就是为什么需要手动触发基础画面的已加载事件

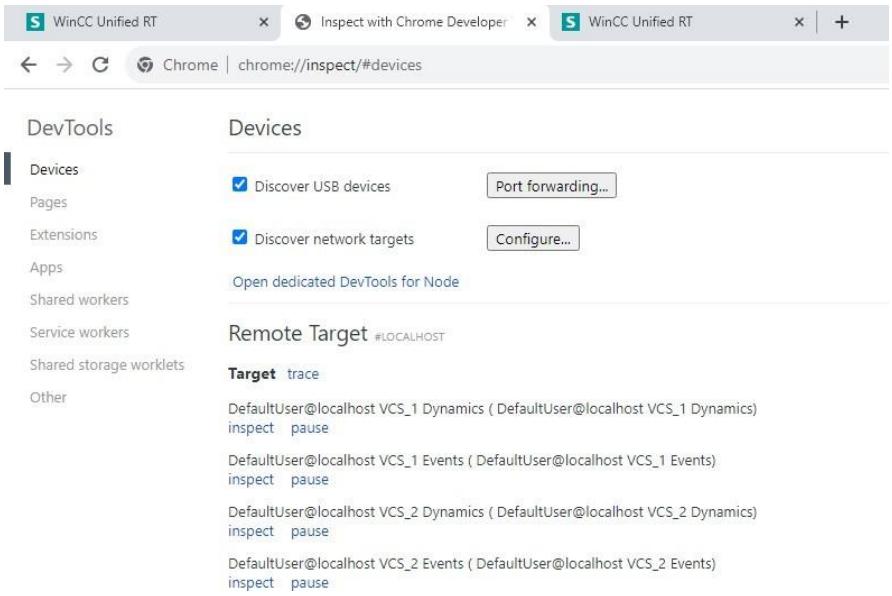


图4-47 选择Runtime 会话的Google Chrome检查页面

5. 附录

5.1. 服务与支持

SiePortal

用于产品选择、采购和支持的集成平台- 连接工业商城和在线支持。SiePortal 主页取代了之前的工业商城和在线支持门户网站 (SIOS) 主页，并将两者结合在一起。

- 产品与服务

  在 "产品与服务"中，您可以找到我们之前在商城目录中提供的所有产品。

- 支持

  在 "支持"页面，您可以找到所有有助于解决产品技术问题的信息。

- mySiePortal

  mySiePortal 收集您的所有个人数据，包括您的账户、当前订单、服务请求等。您只有在登录后才能看到这些全部功能。

您可以通过以下地址访问SiePortal: sieportal.siemens.com

技术支持

西门子工业集团的技术支持部门为您提供量身定制的服务，从基本支持到个体化支持合同等多种类型，就所有技术问题为您提供快速、完备的支持。

请通过网络表单将问题发送至技术支持部门：support.industry.siemens.com/cs/my/src

SITRAIN - 西门子工业技术培训中心

我们为您提供全球范围的行业培训课程，并根据客户的具体需求量身定制方法和概念。

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

工业在线支持应用程序

Available on the App Store

ANDROID APP ON Google play

有了"工业在线支持"应用程序，无论您身在何处，都能获得最佳支持。该应用程序适用于 iOS 和安卓系统：

5.2. 链接和文献

编号	主题
\1\	西门子工业在线支持 https://support.industry.siemens.com
\2\	应用案例的条目页链接 https://support.industry.siemens.com/cs/ww/en/view/109827603
\3\	新手指南：WinCC Unified 的快速入门和转换 https://support.industry.siemens.com/cs/ww/en/view/109810917
\4\	使用 HMI 模板套件进行 HMI 设计 https://support.industry.siemens.com/cs/ww/en/view/91174767
\5\	SIMATIC WinCC Unified 教程中心（视频） https://support.industry.siemens.com/cs/ww/en/view/109782433
\6\	SIMATC WinCC Unified - 使用脚本（JavaScript）的提示与技巧 https://support.industry.siemens.com/cs/ww/en/view/109758536
\7\	TIA-ShowScripts 插件 https://github.com/tia-portal-applications/TIA-Add-In-ShowScripts
\8\	使用Visual Studio Code 开 发WinCC Unified JavaScript 代码并检查风格指南 https://support.industry.siemens.com/cs/ww/en/view/109801600
\9\	SIMATIC HMI WinCC Unified V20 的系统限制 资源监视器 (RT Unified) • WinCC Unified (RT Unified) • Reader • TIA Portal Information System
\10\	SIMATIC WinCC Unified V19 企业设计 SIMATIC WinCC Unified Corporate Designer V19 - ID: 109824234 - Industry Support Siemens

5.3. 文件更改

版本	日期	修改
V1.0	01/2024	第一版
V2.0	04/2024	针对 V19 Upd2 更新
V3.0	01/2025	针对 V20 更新