
## C / C++ 编程器 (C/C++ Developer)

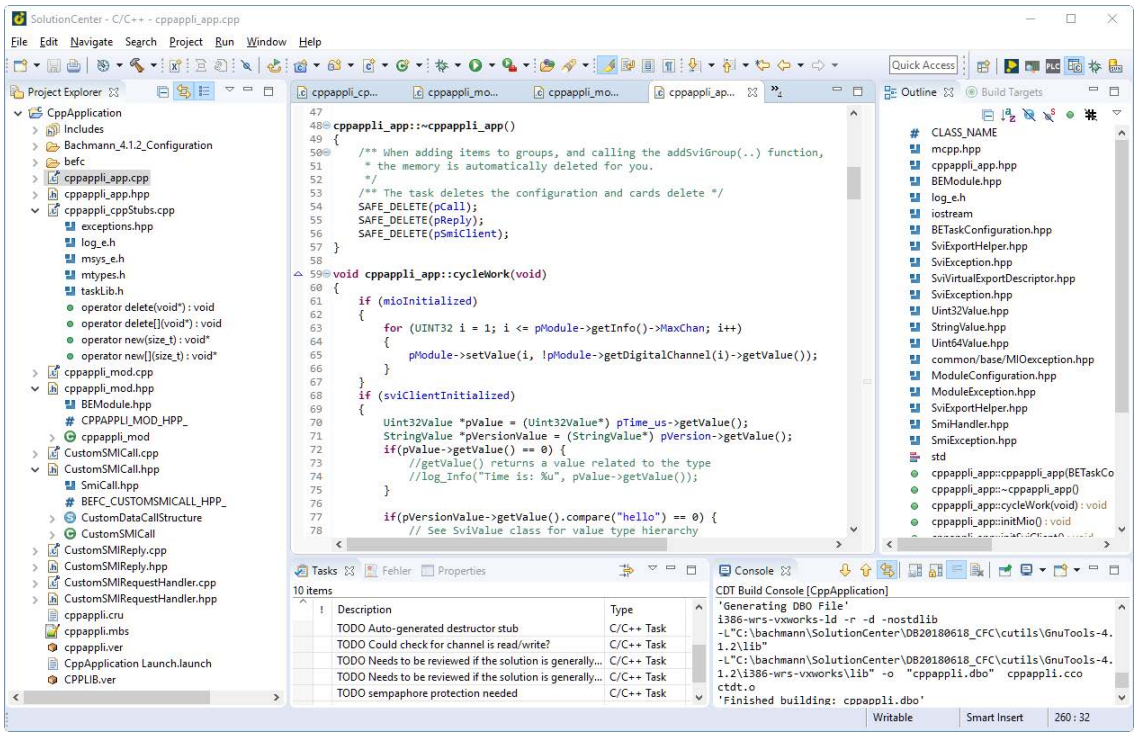
如今，自动化解决的复杂问题，远远超出了单一的设备控制范围。C++ 高级语言编程非常适合处理这些多层任务。它们使得以对象为导向的高效程序编程成为可能。通过 C/C++ Developer 可以得到一个在软件开发的所有阶段，为设备程序员提供全面支持的工具。系统级编程提供了最佳的灵活性和最高的性能。

bachmann.
SolutionCenter

### SolutionCenter 协同效应

SolutionCenter 中集成 C/C++ 为用户提供了创建附加价值所需的协同效应。

- 通过将解决方案项目中的 C/C++ 项目与设备配置和模板相关联，可以将其组合成一个整体的解决方案，并作为一个整体进行管理。
- C/C++ 项目在预先定义的透视图中处理，透视图包含按用户需求布置的编辑器和视图。
- 通过 GIT 和 SVN 的集成版本管理功能，用户可以方便地跟踪代码修改，并为有效的发布管理以及团队合作提供依据。
- 使用集成脚本框架或其他插件形式的扩展（也可以直接从 Eclipse Marketplace 安装）简化了工程和编程工作。



- SolutionCenter 脚本 API 支持应用最新软件开发方法，包括持续集成、模型驱动开发和测试驱动开发等。
- 预先定义的项目模板使用户能够快速开始开发应用。可立即编译的个体特性随后可以选定、转移至控制器并运行。

**高效的编程和管理**

清晰直观的编辑器可实现先进高效的操作。

- 用户定义的语法高亮显示在此处增加了 C 和 C++ 编程语言代码的可读性。
- 得益于完全可用的代码导航和调用层次结构，复杂应用中的相互关系也可以轻松识别。
- 内容相关的自动完成功能和预定义和可扩展的代码和文件模板能够显著提升编程效率。
- 一个工作空间中的项目管理使源文件的所有更改都能够存储在本地历史记录中。无论项目是否进行版本管理，这些变更都可以后续再次调用。
- 代码进行编辑时，广泛的验证功能会即时告知程序员编程错误。
- 通过工具提示调用和显示接口信息和类别与功能文档或变量的文档。
- 项目范围内标识符的更改可以轻松而彻底地执行，就像源代码的格式化一样。

**现代工程**

C/C++ 编程器采用开放、灵活和透明的设计，从而将工程工作推向了一个新的水平。

- 源代码的所有部分都以文本形式保存在项目中，因此用户能够生成整个源代码和配置。因此，可以自动执行重复操作步骤，并防止出现复制/粘贴错误。
- 可以根据需要选择程序部分的结构，从而使应用具有模块化设计。
- 编程的执行与目标平台无关，只需要在执行时选择目标平台即可。编译之后，创建的应用可以直接传输到 M1 控制器并运行。
- 可以为任务设置用户定义的标签，以便更好地查看未决问题。

**通过简单的故障排查提高质量**

C/C++ 编程器的调试框架非常实用，特别是在开发新应用和进行故障排查时。

- 应用可以通过断点在任何位置停止，然后分步执行。可以联合管理、激活或完全停用断点。
- 可以将条件分配给断点，以便在特定情况下停止应用运行。
- 应用停止时将显示堆栈框架，通过该框架还可以导航到要调用的功能。
- 还可以同时调试多个应用和任务。
- 反汇编视图可实现汇编程序代码中的监控和调试。源代码同时显示。

| C/C++ 编程器 | |
|---|---|
| **总述** | |
| 集成 | • 在一个工作空间内管理多个 C/C++ 项目<br>• 在一个工作空间内管理所有设备和 C/C++ 项目<br>• C/C++ 项目在解决方案项目中关联，用于联合管理 |
| 显示 | • 为处理 C/C++ 项目预定义的透视图<br>• 视图和编辑器的可自由选择且清晰的布置 |
| 版本管理 | • SVN（子版本）<br>• GIT（本地和远程） |
| 自动化 | • SolutionCenter 中运行脚本（JavaScript 和 Python）<br>• 用于创建和传输 C/C++ 应用的预定义脚本功能 |
| 可扩展性 | • 可通过 Eclipse Marketplace 安装的扩展（插件）<br>• 使用用户创建的 Eclipse 插件 |
| 可扩展性 | • 独立于目标平台创建应用<br>• 所有 M200-CPU 上的传输和执行 |
| 结构 | • 工作集中的项目组合<br>• 可在项目中自由定义的文件夹结构<br>• 可全面访问所有源代码文件 |
| 模板 | • 带有功能代码片段的预定义项目模板<br>• PLC 库模板<br>• C 组件模板<br>• 自定义项目模板 |
| **编辑器** | |
| 语言 | • 符合 IEC 9899:1999 的 C 语言 (C99)<br>• 符合 IEC 14882:2011 的 C++ 语言 (C++11)<br>• 符合 IEC 14882:2014 的 C++ 语言 (C++14) |
| 代码导航 | • （在项目中）查找引用<br>• 调用层次结构<br>• 调用者层次结构（可扩展的层次结构）<br>• 开放声明 |
| 自动完成 | • 内容相关（声明，方法）<br>• 预定义的源代码模板<br>• 用户定义的源代码模板 |
| 语法高亮度显示 | • 预定义<br>• 可自由配置 |
| 变更跟踪 | • 与本地存储版本的比较（本地历史记录）<br>• 与 SVN 或 GIT 的源代码比较 |
| 变更 | • 标识符的全项目重命名<br>• 源代码格式化 |
| 信息 | • 标识符工具提示中的内容相关信息 |
| 验证 | • 文档中系统功能的描述<br>• 在源代码和错误视图中直接显示问题<br>• 对源代码进行语法和语义检查<br>• 静态代码分析 |
| 任务 | • 在注释中记录未决问题<br>• 在单独视图中显示所有任务<br>• 自定义标签的定义 |

| C/C++ 编程器 | |
|---|---|
| **配置** | |
| 模块配置 | • 通过配置描述进行定义（CRU）<br>• 用于调整模块配置的编辑器 |
| 构建配置 | • 一个项目中定义多个构建配置<br>• 同时使用多个构建配置进行构建 |
| **编译** | |
| 工具链 | • GCC 4.1.2 (已包含)<br>• GCC 5.5 (已包含)<br>• MinGW<br>• Cygwin |
| 构建 | • 手动<br>• 自动<br>• 可以定义构建前和构建后的步骤<br>• 在多处理器内核上并行构建 |
| **在线操作** | |
| 安装 | • 目标控制器永久或动态可选<br>• 临时或永久安装在目标控制器上<br>• 模块实例的名称可自由定义<br>• 连接到正在运行的应用（附加） |
| 初始化 | • 可以开始进行定义的调试 |
| 部署配置 | • 每个项目可以有多个配置<br>• 可以组合执行多项配置（启动组）<br>• 收藏夹管理 |
| **调试 - 故障排查** | |
| 按步处理 | • 通过断点停止应用<br>• 步入/超过/返回<br>• 条件断点<br>• 汇编程序步进 |
| 监控 | • 停止应用时的变量值<br>• 带有汇编指令和源代码的反汇编视图 |
| 显示 | • 更改 INT 值的格式（dec、hex、bin 或 oct）<br>• 应用停止时的当前堆栈帧 |
| 操作 | • 可以设置变量值 |