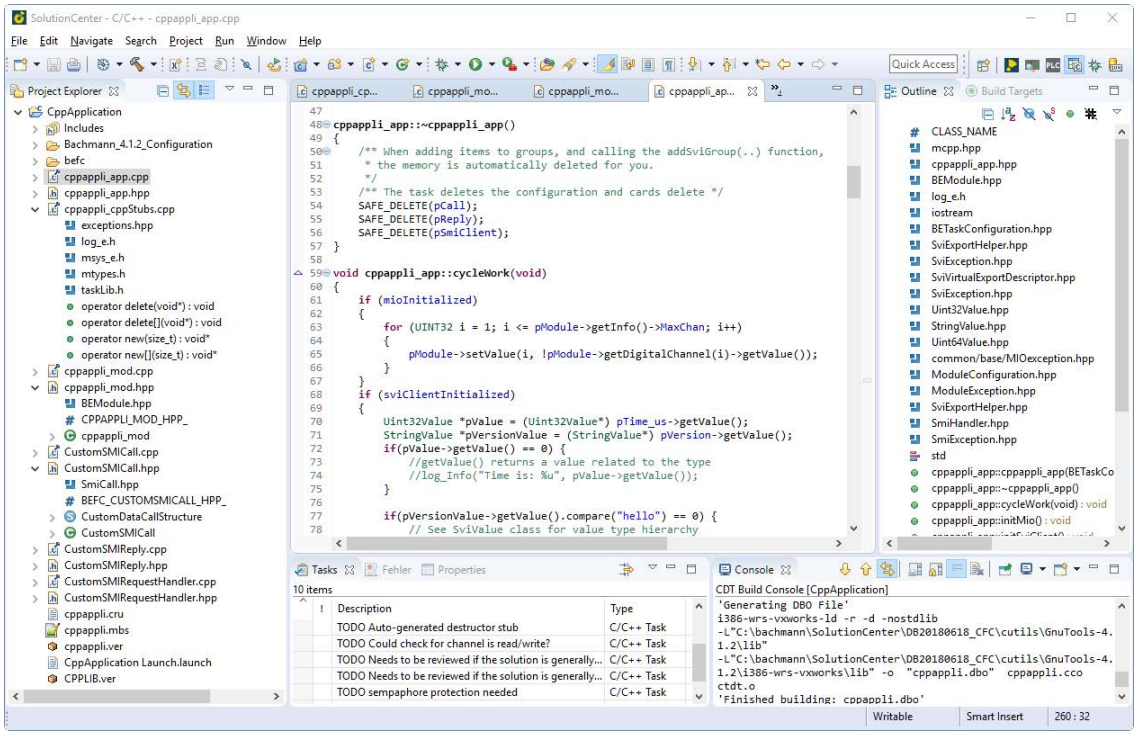
C / C++ Developer

Modern automation systems solve complex tasks that go far beyond the controlling of a plant. Procedural or object-oriented programming in C and C++ languages makes it possible to solve a wide range of tasks in an efficient and structured way. The C/C++ Developer supports the system programmer here through all phases of software development. System-level programming offers optimum flexibility with maximum performance.

Synergies in the SolutionCenter

The integration of the C/C++ Developer in the SolutionCenter gives users the synergies needed to create added value.

- By linking C/C++ projects in a Solution project with device configurations and templates, they can be combined to form a complete solution and managed as one.
- The C/C++ projects are processed in a predefined perspective containing editors and views arranged to meet the needs of the user.
- The integrated version management function using GIT and SVN enables the user to track code modifications easily and provides the basis for effective release management as well as working together in teams.
- The use of the integrated scripting framework or other extensions in the form of plugins, which can also be installed directly from the Eclipse Marketplace, simplifies engineering and programming.



- The use of SolutionCenter scripting API enables the application of modern software development methods such as continuous integration, model-driven development and test-driven development.
- Predefined project templates enable users to get started quickly in developing applications. Individual features, which can be compiled immediately, can then be selected, transferred to the controller and run.

Efficient programming and management
The clear and intuitive editors allow modern and efficient operation.

- The user-defined syntax highlighting increases here the readability of code in C and C++ programming languages.
- Thanks to the fully available code navigation and call hierarchy, it is also possible to easily identify interrelationships in complex applications.
- Efficient programming is particularly enhanced by the context-sensitive autocomplete function and the predefined and extendable code and file templates.
- Project management in a workspace enables all changes to the source files to be stored in a local history. These can be called again at a later time irrespective of whether the project is version managed.
- An extensive validation function immediately notifies the programmer of programming errors when the code is edited.
- Information on interfaces and the documentation of classes and functions or variables can be called and displayed via tooltips without leaving the editor.
- The project-wide changing of identifiers can be carried out easily and thoroughly, just like the formatting of source codes.

Modern engineering
The C/C++ Developer has an open, flexible and transparent design, and thus brings the engineering to a new level.

- All sections of the source code are saved in text form in the project, thus enabling the user to generate entire source codes and configurations. Recurring operating steps can thus be automated and copy/paste errors prevented.
- The structuring of the program sections can be selected as required and thus enables the application to have a modular design.
- Programming is carried out irrespective of the target platform, which only has to be selected at the time of execution. After compilation, the created application can be transferred directly to an M1 controller and run.
- User-defined tags can be set for tasks in order to keep open issues better in view.

Higher quality through simple troubleshooting
The debug framework of the C/C++ Developer is extremely useful, particularly with the development of new applications and with troubleshooting.

- An application can be stopped at any position via a breakpoint and then executed in steps. Breakpoints can be managed jointly and activated or deactivated altogether.
- Conditions can be assigned to breakpoints so that the application can be stopped in specific circumstances.
- When the application is stopped, the stack frame is displayed by which it is also possible to navigate to the functions called.
- It is also possible to debug several applications and tasks simultaneously.
- The Disassembly view enables monitoring and debugging to be carried out in the assembler code. The source code is shown at the same time.

C/C++ Developer	
General	
Integration	• Management of several C/C++ projects in a workspace • Management of all devices and C/C++ projects in a workspace • Linking of C/C++ projects in the Solution projects for joint management
Display	• Predefined perspectives with views for processing C/C++ projects • Clear, freely selectable arrangement of views and editors
Version management	• SVN (subversion) • GIT (local and remote)
Automation	• Running scripts in the SolutionCenter (JavaScript, Python) • Predefined script functions for creating and transferring C/C++ applications
Expandability	• Extensions (plugins) installable via Eclipse Marketplace • Use of user-created Eclipse plugins
Scalability	• Creation of the application independent of the target platform • Transfer and execution on all M200-CPUs
Structure	• Combination of projects in working sets • Folder structure freely definable in the project • Full access to all source code files
Templates	• Predefined project templates with functional code snippets • Templates for PLC libraries • Templates for C components • Custom project templates
Editors	
Languages	• C according to IEC 9899:1999 (C99) • C++ according to IEC 14882:2011 (C++11) • C++ according to IEC 14882:2014 (C++14)
Code navigation	• Finding references (in the project) • Call hierarchy • Caller hierarchy (hierarchy levels expandable) • Opening declaration
Auto-complete	• Context-sensitive (declaration, methods) • Predefined source code templates • User-defined source code templates
Syntax highlighting	• Predefined • Freely configurable
Change tracking	• Comparison with locally stored versions (local history) • Comparison with source code from SVN or GIT
Changes	• Project-wide renaming of identifiers • Formatting the source code
Information	• Context-sensitive information in the tooltip on identifier
Validation	• Description of the system functions in the documentation • Display of problems directly in the source code and in the error view • Syntactical and semantic checking of the source code • Static code analysis
Tasks	• Documentation of open issues in comments • Display of all tasks in a separate view • Definition of custom tags

C/C++ Developer	
Configuration	
Module configuration	• Definition via configuration description (CRU) • Editor for adjusting the module configuration
Build configuration	• Definition of multiple build configurations in one project • Build with multiple build configurations at the same time
Compiling	
Toolchain	• GCC 4.1.2 (already included) • GCC 5.5 (already included) • MinGW • Cygwin
Build	• Manually • Automatically • Pre-/post-build steps can be defined • Parallel build on multiple processor cores
Online Operation	
Installation	• Target controller permanently or dynamically selectable • Installation on target controller temporarily or permanently • Name of the module instance freely definable • Connect to running application (attach)
Initialization	• Defined debug start possible
Deploy configurations	• Several configurations per project possible • Combined execution of several configurations possible (launch groups) • Management of favorites
Debugging – Troubleshooting	
Stepwise processing	• Stopping of the application via breakpoint • Step into/over/return • Conditional breakpoints • Assembler stepping
Monitoring	• Variable values when the application is stopped • Disassembly view with assembler instructions and source code
Display	• Changing the format of INT values (dec, hex, bin, oct) • Current stack frame when the application is stopped
Manipulation	• Variable values can be set