bachmann.
SolutionCenter

C/C++ Developer

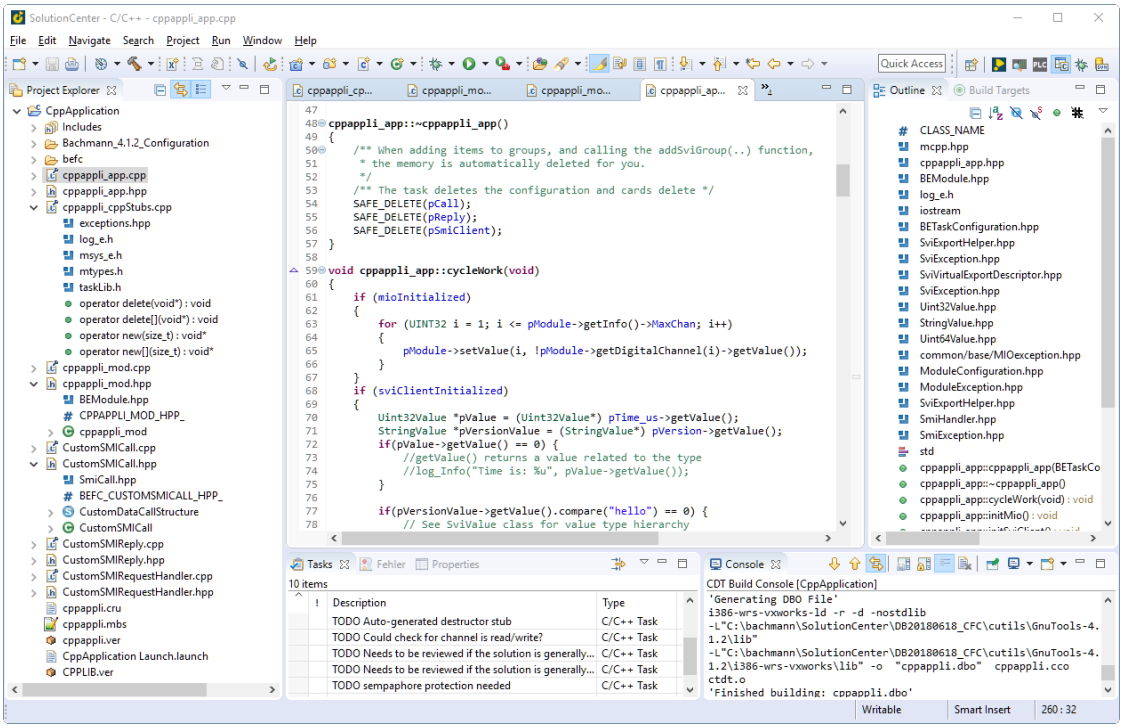
Current automation systems solve complex problems that go well beyond merely controlling the system. Procedural or object-oriented programming in C and C++ languages offers the possibility to solve different tasks in an efficient and structured way. The C/C++ Developer supports the system programmer in all phases of software development. System-level programming offers optimum flexibility with maximum performance.

Synergies in the SolutionCenter

Because of the integration of the C/C++ Developer in the SolutionCenter synergies can be exploited and therefore added value can be created.

Features

- 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 a 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.


SolutionCenter - C/C++ - cppappli_app.cpp
File Edit Navigate Search Project Run Window Help
Project Explorer
CppApplication
Includes
Bachmann_4.1.2_Configuration
befc
cppappli_app.cpp
cppappli_app.hpp
cppappli_cppStubs.cpp
exceptions.hpp
log_e.h
msys_e.h
mtypes.h
taskLib.h
operator delete(void*) : void
operator delete[](void*) : void
operator new(size_t) : void*
operator new[](size_t) : void*
cppappli_mod.cpp
cppappli_mod.hpp
BEModule.hpp
CPPAPPLI_MOD_HPP_
cppappli_mod
CustomSMICall.cpp
CustomSMICall.hpp
SmiCall.hpp
BEFC_CUSTOMSMICALL_HPP_
CustomDataCallStructure
CustomSMICall
CustomSMIReply.cpp
CustomSMIReply.hpp
CustomSMIRequestHandler.cpp
CustomSMIRequestHandler.hpp
cppappli.cru
cppappli.mbs
cppappli.ver
CppApplication Launch.launch
CPPLIB.ver
cppappli_cp...
cppappli_mo...
cppappli_mo...
cppappli_ap...
cppappli_app::~cppappli_app()
{
/** When adding items to groups, and calling the addSviGroup(..) function,
* the memory is automatically deleted for you.
*/
/** The task deletes the configuration and cards delete */
SAFE_DELETE(pCall);
SAFE_DELETE(pReply);
SAFE_DELETE(pSmiClient);
}
void cppappli_app::cycleWork(void)
{
if (mioInitialized)
{
for (UINT32 i = 1; i <= pModule->getInfo()->MaxChan; i++)
{
pModule->setValue(i, !pModule->getDigitalChannel(i)->getValue());
}
}
if (sviClientInitialized)
{
Uint32Value *pValue = (Uint32Value*) pTime_us->getValue();
StringValue *pVersionValue = (StringValue*) pVersion->getValue();
if(pValue->getValue() == 0) {
//getValue() returns a value related to the type
//log_Info("Time is: %u", pValue->getValue());
}
if(pVersionValue->getValue().compare("hello") == 0) {
// See SviValue class for value type hierarchy
Outline
Build Targets
CLASS_NAME
mcpp.hpp
cppappli_app.hpp
BEModule.hpp
log_e.h
iostream
BETaskConfiguration.hpp
SviExportHelper.hpp
SviException.hpp
SviVirtualExportDescriptor.hpp
SviException.hpp
Uint32Value.hpp
StringValue.hpp
Uint64Value.hpp
common/base/MIOexception.hpp
ModuleConfiguration.hpp
ModuleException.hpp
SviExportHelper.hpp
SmiHandler.hpp
SmiException.hpp
std
cppappli_app::cppappli_app(BETaskCo
cppappli_app::~cppappli_app()
cppappli_app::cycleWork(void) : void
cppappli_app::initMio() : void
Tasks
Fehler
Properties
10 items
Description
Type
TODO Auto-generated destructor stub
C/C++ Task
TODO Could check for channel is read/write?
C/C++ Task
TODO Needs to be reviewed if the solution is generally...
C/C++ Task
TODO Needs to be reviewed if the solution is generally...
C/C++ Task
TODO sempaphore protection needed
C/C++ Task
Console
CDT Build Console [CppApplication]
'Generating DBO File'
i386-wrs-vxworks-ld -r -d -nostdlib
-L"C:\bachmann\SolutionCenter\DB20180618_CFC\cutils\GnuTools-4.1.2\lib"
-L"C:\bachmann\SolutionCenter\DB20180618_CFC\cutils\GnuTools-4.1.2\i386-wrs-vxworks\lib" -o "cppappli.dbo" cppappli.cco ctdt.o
'Finished building: cppappli.dbo'
Writable
Smart Insert
260 : 32

Efficient programming and management

The clear and intuitive editors allow a modern and efficient way of working.

- The user-defined syntax highlighting increases 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 auto-completion 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 protocol. 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, 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 is open, flexible and transparent designed to take engineering to a new level.

- All sections of the source code are stored in text form in the project, which enables the user to generate entire source files and configurations. Recurring operating steps can thus be automated and copy and paste errors prevented.
- The structuring of the program sections can be selected as required and thus enables the application program 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 program can be transferred directly to an M200 controller and run.
- User-defined tags can be set for tasks in order to keep a better overview of open issues.

Higher quality through simple troubleshooting

The debug framework of the C/C++ Developer is extremely helpful during the development of new application programs and to find errors.

- An application program 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 to stop the application program under certain circumstances.
- When the application program is stopped, the stack frame is displayed by which it is also possible to navigate to the functions called.
- It is possible to debug several application programs and tasks simultaneously.
- The Disassembly view permits 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
Revision control	• SVN (Subversion) • GIT (local und remote)
Automation	• Running scripts in the SolutionCenter (JavaScript, Python) • Predefined script functions for creating and transferring C/C++ application programs
Expandability	• Extensions (plugins) installable via Eclipse Marketplace • Use of user-created Eclipse plugins
Scalability	• Creation of the application programs 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 • User-defined project templates
Editors	
Languages	• C acc. IEC 9899:1999 (C99) • C++ acc. IEC 14882:2011 (C++11) • C++ acc. IEC 14882:2014 (C++14)
Code navigation	• Find references (in the project) • Call hierarchy • Caller hierarchy (hierarchy levels expandable) • Open 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 procotol) • 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
Configuration	
Module configuration	• Definition via configuration description (CRU) • Editor for adjusting the module configuration

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	• Manual • Automatic • 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 program (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 program via breakpoint • Step into/over/return • Conditional breakpoints • Assembler stepping
Monitoring	• Variable values (when the application program 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 program is stopped)
Manipulation	• Setting variable values