Combining Your Ada & C/C++ Projects

Ada-C/C++ CHANGER®

Convert your project.
Reuse your design.
Simplify your maintenance.

Integrate Your Projects

MAPUSOFT

THE GLOBAL LEADER
IN SOFTWARE INTER-OPERABILITY

Ada-C/C++ CHANGER®

Table of Contents

Executive Summary ..... Page 3

Background and History

Combining Ada and C/C++ projects ..... Page 4
History of Ada ..... Page 4
History of C/C++ ..... Page 5
The debate between two superhero languages ..... Page 6
And the winner is... ? ..... Page 7
Special Section: A Note to the Language Guys ..... Page 7

The Problem

Do I have to abandon my project? ..... Page 8
Why should I combine my projects? ..... Page 8
Are there really projects combining C/C++ and Ada? ..... Page 10
What are the real costs for development? ..... Page 10

The Solution – Ada-C/C++ Changer

How do you combine projects? ..... Page 11
Setting up the environment – Easy and Intuitive ..... Page 11
Supports the simple, but handles the complex projects ..... Page 13
You can develop in Ada, but debug in C/C++ ..... Page 13
Integrating with a real-time operating system ..... Page 14
Support for multiple targets, operating systems, and interfaces ..... Page 15
For other configurations ..... Page 16
Conclusion ..... Page 17

About Us and Contact

About Mapusoft ..... Page 18
About Ada-C/C++ Changer ..... Page 18
For more information ..... Page 18
Contact Us ..... Page 19

Ada-C/C++ CHANGER®

Executive Summary

There is an ongoing debate among programmers using Ada and C/C++. The Ada developers think that C or C++ is a dangerous and poorly designed language. Also, since the government issued a mandate in 1987 for all DoD projects to be developed and written in Ada, there is a certain amount of pride surrounding this language. While the mandate existed, there were thousands of projects developed and successfully launched using Ada. Those projects amounted to hundreds of millions of lines of code.

There is no clear winner between the languages. Both languages have strengths. Ada has been used for decades for government and DoD development. C/C++ is an incredibly powerful language in the hands of an experienced programmer. There is no clear winner in the debate.

There is one thing that is certain: future development will be focused around the integration of Ada projects and C/C++ projects. There are too many existing Ada applications still in the field for the language to disappear, yet most new projects are using C or C++ as the language of choice.

The solution to integrating two projects in Ada and C/C++ is to use Mapusoft's Ada-C/C++ Changer. Ada-C/C++ Changer easily converts software written in Ada code to C/C++, allowing Ada applications to be combined with C/C++ projects. The resultant C/C++ software can be integrated with OS Abstractor® from Mapusoft's Cross-OS Development Platform to support a wide variety of host and target OS platforms. This conversion process eliminates the need for costly and tedious code rewrites, saving you time and money.

Ada-C/C++ Changer is a fully automatic conversion engine and requires no human intervention. It converts 100% of your Ada source into C/C++. The Ada-C/C++ Changer includes a fully validated Ada compiler, which supports Ada95 and parts of Ada 2005. The output code is efficient and readable C/C++ that exactly matches the semantics of the original Ada program. The Ada-C/C++ Changer also includes a GNAT Ada compatibility option to allow you to easily convert Ada code developed utilizing GNAT compilers. It also includes a POSIX Ada interface package to assist you in transitioning your code that relies on POSIX features.

Combining Ada and C/C++ projects

For government projects, the idea that most software projects involve entirely new designs is wrong. The government, particularly the Department of Defense, has been writing software for over 50 years. Although not all of that software is still in operation, there are millions of lines of code that are. When a new development project comes up for review, there will likely be new software development as well as software integration with an existing project. In this paper, we will examine the idea of integrating software projects using two very popular languages in the embedded market – Ada and C/C++. This paper will also present a tool – Mapusoft's Ada-C/C++ Changer – that simplifies the process of integrating the two languages.

History of Ada

The Ada language has an interesting lineage. Officially released in the late 1970s, it has been through several revisions over the last few decades. It has been used for millions of lines of code, mostly in government related projects. The most widely used revision, Ada95, was released in 1995. But the start of the language and its role in government projects is interesting.

In the early 1970s, the Department of Defense was spending significant time, money, and resources on software development. Most of the development was on embedded applications. One study estimated that in 1973-1974, around $3Billion was spent on software development. (In today's terms, that's around $14.5Billion.) Unfortunately at the time there was no common language or hardware platform in existence. For projects going on during this period, over 450 different languages were used. Most of these languages were outdated, cumbersome, or not portable enough to be used across multiple platforms. To reduce this complexity, a common high-level language needed to be developed.

A working group was formed in 1975 to design a new language that could act as a standardized language for many of the DoD's projects. After much analysis and several proposals, a single proposal was chosen for approval in 1979. The language was named after Ada Lovelace (1815-1852), who worked on Charles Babbage's early mechanical computer, thus being considered the first computer programmer.

From there, the language grew in popularity within new government projects. As the popularity of Ada grew, the government issued a specific mandate in 1987 to make sure that all new development was done in Ada.

The language has gone through several revisions since its initial creation. It became an ANSI standard in 1983 and an ISO Standard in 1987. Ada95, the most widely used version of the standard, was officially published in 1995, with an update released in 2001. Another version of Ada was published in 2007.

To date, Ada is still quite entrenched in the embedded world, especially in ongoing government software projects. Although the mandate for using Ada was lifted by the U.S. government in 1997, there remains a large installed base of Ada applications. While there only a small fraction of new projects are being started with Ada as the language of choice, many software projects today are updating existing Ada applications and integrating with other software projects written in other languages, such as the C programming language.

History of C/C++

The programming language C started as a project for Bell Laboratories by Dennis Ritchie. The language was developed in the early 70s alongside the development of the Unix operating system. It was developed as a way to reduce the many complexities and portability issues related to software development using assembly language. At the time, many software projects were written in assembly language.

Like many applications at the time, the UNIX operating system was originally written in assembly language. As C was introduced and formalized, Unix was rewritten using C. The language itself provided low-level access like assembly, but offered a higher level syntax. This eased the burden on software developers and made it easier to write code and develop applications. This flexibility and focus on portability provided a great language for development of system software, which fueled the growth of the embedded software market.

For embedded systems, the popularity of C grew based on it's flexibility as a language. As the embedded software market grew, the number of available semiconductors exploded and C became the de facto standard. There are very few 32-bit processors, if any, that don't have some type of support for the C language.

Another important aspect governing the growth of embedded systems and the C language has to do with its modularity and portability. Certain functions can be written in C and are portable across many different architectures. Because of this, entire libraries of C functions can be written and stored in libraries for use across multiple platforms. This reduces the amount of re-writing for a programmer using C in their software development. The rise of open source software has also helped fuel the growth of C. As more code is released into the community, developers continue to update and contribute to the code base, creating an ever-increasing pool of working software.

In 1989, a version of the C language was ratified by American National Standards Institute (ANSI), creating the first version of ANSI C. That standard was adopted

as an ISO standard in 1990. There have been several minor updates to the specification since then, but the language remains very close to its original form.

In 1979, Bjarne Stroustrup took the C language and added several enhancements to the language. In 1983, this new language was named C++. Since that time, it has seen dramatic growth within the software industry. Today it is a very popular programming language for software developers. Many projects use a combination of both the C language and the C++ language in their development.

Today's embedded marketplace, C and C++ are the most popular and dominant languages being used for development. Because of its popularity and history, we have a great number of software developers with strong C/C++ language skills available in the market today.

Why so serious?
The debate between two superhero languages

There is an ongoing debate among programmers using Ada and C/C++. The Ada developers think that C or C++ is a dangerous and poorly designed language. Also, since the government issued a mandate in 1987 for all DoD projects to be developed and written in Ada, there is a certain amount of pride surrounding this language. While the mandate existed, there were thousands of projects developed and successfully launched using Ada. Those projects amounted to hundreds of millions of lines of code.

The development of Ada was through rigorous design and requirements documents over many years. It went through major revisions and has reached a level of stability and maturity that many languages never see. And it was designed for real-time and embedded systems. So, in some sense, Ada is the premier language for real-time embedded system development.

On the other hand, C/C++ developers believe Ada is a language with little future. The new processors and tools coming out in the market are all guaranteed to have support for C or C++. C and C++ as languages have extensive demo code, libraries, sample applications, and tools that facilitate the development process. The C /C++ compilers are continually being updated. In addition, C/C++ offers incredible flexibility and powerful access to the low level features of the processor, providing lots of capability during the application development process. In the mind of some developers, C/C++ is the language of choice.

Ada-C/C++ CHANGER®

And the winner is… ?

No one.

There is no clear winner. Both languages have strengths. Ada has been used for decades for government and DoD development. C or C++ is an incredibly powerful language in the hands of an experienced programmer and widely accepted by the industry. There is no clear winner in the debate.

There is one thing that is certain: future development will be focused around the integration of Ada projects and C/C++ projects.

Note to the language guys

Look guys… Here's the deal -

If you've been a programmer for any length of time, you know that the progression of technology – compilers, semiconductors, operating systems, and programming languages – rises and falls. Over time, there is an ebb and flow for all things technological.

While each facet of technology has its own strengths and weaknesses, sometimes there are people who are adamant and fierce proponents of one solution over the others. In this particular situation, there are often arguments surrounding the use of Ada and C/C++.

MapuSoft does not advocate the use of one programming language over the other. We are simply stating the fact that in today's development environment there is a growing need to combine Ada code and C/C++ code. This is an almost self-evident future growth trend.

The maturity and lifecycle of a technology product is determined by a multitude of factors – product maturity, marketplace acceptance, government awareness, capitalistic mechanisms, and symbiotic technology. All of those things are out of your (and our) control. Our experience (and this paper) is based on our exposure in the marketplace, discussions with our customers, discussion with prospects, marketplace/industry data, and discussions with leading experts.

At MapuSoft, our goal is not to direct or suggest the language you use, only to make your life easier after that choice has been made.

Do I have to abandon my project?

By far, the biggest misconception you may have about combining projects is that you need to abandon everything you've ever written and move to C/C++.

In fact, as we'll discuss later, you can actually continue developing in Ada and then test your code in C/C++. There is absolutely no reason to cause you to abandon your existing work.

There are two very legitimate reasons why you should not abandon an existing project – money and time. If you've spent a week writing code, that represents an investment of your time and effort. You expend that effort on your employer's behalf in order to complete the project. And your employer pays you in return for your time. The money and time are both invested in order to reach a finished project.

To throw away those two components would be wasteful and inefficient. The much better solution is to figure out a way to leverage your existing work and to re-use as much code as you can.

Why should I combine my projects?

There are several reasons why you would want to combine projects written in Ada and C/C++. Here are a few:

Retooling/Rehosting/Rework is expensive

Sometimes when you start to update an existing software application, the ancillary costs start to escalate. For example, you may need to upgrade your host machine, development environments, licenses, hardware platforms, compilers, etc. The overall project costs can grow outrageously expensive. In some cases the rehosting expense can be greater than the cost of the original development.

In addition to the high costs of rehosting, there are also other costs. It takes time to set up and configure new machines and new tools. It takes time to start up new boards and make sure they are working properly. When you consider the amount of time needed to configure and set up the development environment, the overall cost to the development schedule will be substantial. These additional costs are hard to justify for new projects, especially in today's economic atmosphere and budget cuts.

Lack of programmers

Because of the proliferation of C/C++ programmers, there may be a lack of Ada developers in the market. Depending on the application, you may need to hire additional expertise in order to maintain the current software. Or, you may need to hire new programmers to learn the existing codebase. With Ada, this effort will be much harder than that needed to hire C/C++ programmers.

Existing sample code / Support for new platforms

The idea that demo code can influence a project may sound silly, but it happens. Demo code can be a simple "Hello World" example, but it can sometimes be complex code routines that take advantage of advanced functionality of a processor. Those code samples may range from the initialization of the chip to properly configuring interrupt controllers or registers. The code could come from the community at large or it could come from the chip supplier. In today's embedded software market, most likely this code is written in C.

Availability of middleware

In many situations, a software application incorporates multiple middleware packages. This middleware can come from public sources or it can come from other vendors. While there exist middleware packages for Ada, the number of middleware packages written in C/C++ are more diverse. Additionally, the total number of middleware packages available in C/C++ outnumber the ones available for Ada development.

Government mandate

In recent years, there has been a push from government agencies to encourage and recommend projects that are re-using existing code. The concept behind leveraging existing code and current software applications has been mentioned throughout this paper. The idea of re-use makes absolute sense from a software perspective, but there is now active support from government officials on combining projects and reusing existing code.

Are there really projects combining C/C++ and Ada?

Recently, there has been a combination of two government projects that were asked to integrate into a single project. Both of these projects are well-known simulator applications, with one that has been running approximately 20 years and the other one operating for 10, one in Ada, and one in C++.

The directive? Combine both of those projects into a single simulator.

The reason? By combining both projects, the overall development costs will go down dramatically and the functionality of the simulator will be improved by leveraging mutual functionality and deepening the learning curve on both projects.

So, yes, this is happening today.

What are the real costs for development?

Development costs for writing an application in Ada can vary depending on the type of application. The development cost per line of code jumps significantly if the application is used in a safety-critical system. This increase is due to the added cost of testing. If you are developing a high-security application, those numbers go even higher.

In practice, the range of developing an application in Ada is $25-$50 per line of code. When you factor in the rigorous testing for a safety-critical application, the cost per line doubles to $100 for each line written.

If you are translating code from Ada to C, the costs average around $5 per line for the translation and then an additional $10 per line for testing. The real challenge comes from finding the right engineer to make the translation happen. If you were to handle the translation by hand, you would need an engineering team proficient in both Ada and in C. That's where the Ada-C/C++ Changer makes the difference.

Ada-C/C++ Changer reduces the laborious task of manual translation. Rather than taking several months to accomplish the translation from Ada to C, using Ada-C/C++ Changer condenses that time down to a few hours. Also, the Ada-C/C++ Changer does the translation perfectly, so you preserve the semantics of the original Ada program. This allows you to cut down on the time spent testing, since you can use the exact same test suite and receive identical output.

Ada-C/C++ CHANGER®

How do you combine projects? Here's how:

The solution to integrating two projects in Ada and C/C++ is to use Mapusoft's Ada-C/C++ Changer. Ada-C/C++ Changer easily convert's software written in Ada code to C/C++ thereby allowing Ada applications to be combined with C/C++ projects. The resultant C/C++ software can be integrated with Mapusoft's OS Abstractor® environment to support a wide variety of host and target OS platforms. This conversion process eliminates the need for costly and tedious code rewrites, saving you time and money.

Ada-C/C++ Changer is a fully automatic conversion engine and requires no human intervention. It converts 100% of your Ada source into C/C++. The Ada-C/C++ Changer includes a fully validated Ada compiler, which supports Ada95 and parts of Ada 2005. The output code is efficient and readable C/C++ that exactly matches the semantics of the original Ada program. The Ada-C/C++ Changer also includes a GNAT Ada compatibility option to allow you to easily convert Ada code developed utilizing GNAT compilers. It also includes a POSIX Ada interface package to assist you in transitioning your code that relies on POSIX features.

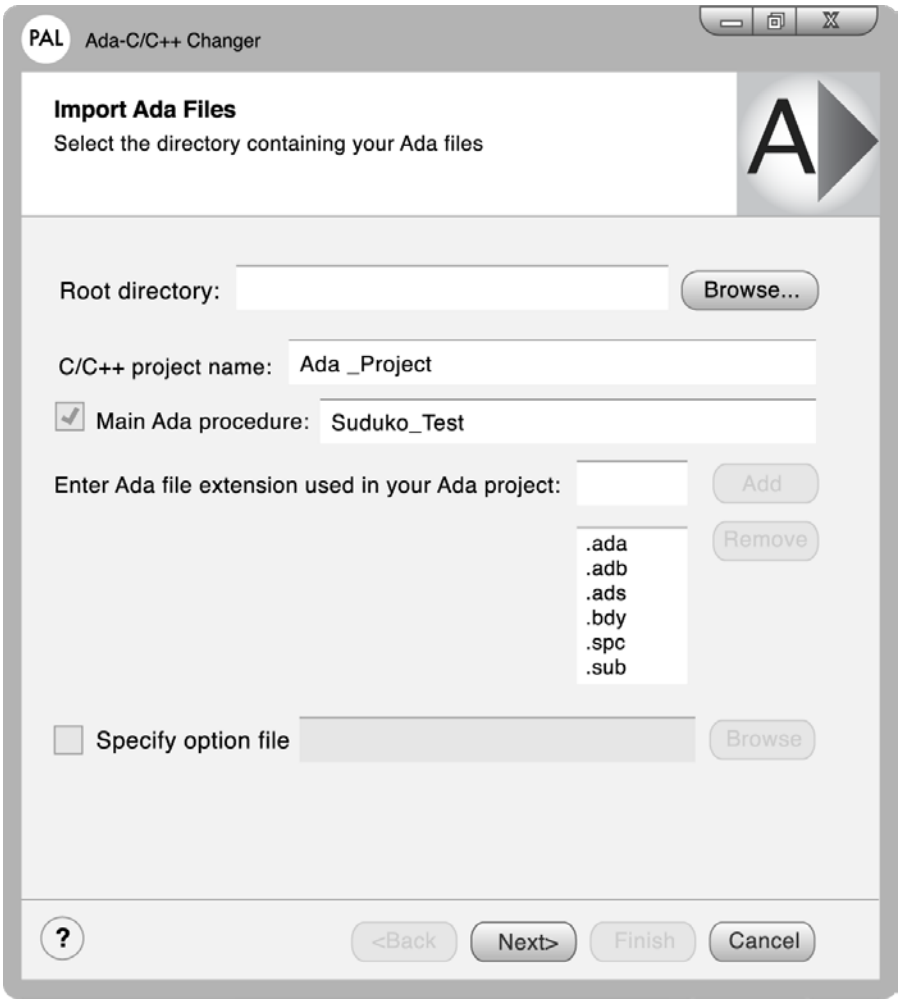
Setting up the environment – Easy and Intuitive

Setting up the Ada-C/C++ Changer is straightforward for all projects. It is designed to work on all types of projects, large, small, simple and complex. The interface is a simple set of dialog boxes that guide you through the entire setup process.

With the Changer, there are no constraints on directory structure, file layout, or configuration management. So however your files are organized, or whichever configuration management tool you are using, the Ada-C/C++ Changer will work with it.

Additionally, it provides complete freedom in regards to your file naming conventions. There are no limitations on naming conventions and extensions of your source files. You can use the common extensions for your files - .ada, .adb, .ads, .bdy, .spc, .sub. – or you can define your own custom extension within the tool.

Ada-C/C++ Changer also includes C/C++ run-time sources that provide I/O, tasking, exception handling, and memory management modules. These functions are normally required by the Ada 95 language, to be called by the C/C++ code that has been converted. These are referred to as the Ada run time library (RTL)



The tool also makes it easy for the programmer to transition from viewing Ada code to viewing C/C++ code. This is done by carefully maintaining a solid consistency during the code conversion process. Ada-C/C++ Changer preserves Ada code's comments, files structures, and variable names during the transition to C/C++ code. This makes it simple for a programmer to understand and comprehend the two software applications even though they are in two separate languages.

Ada-C/C++ Changer also tracks changes to the source files, so you save time during the build process by only compiling the files that have been modified since the last build.

Supports the simple, but handles the complex projects…

The default setting for the Ada/C-C++ tools is for small projects. The default settings allow you to immediately start working on code and running demos. The defaults are there for you to get up and running quickly and to start testing demos and simple applications.

Ada/C-C++ Changer can be configured for larger, more complex projects. Many large projects, especially government projects, will involve multiple Ada or C/C++ source code directories and libraries. These directories may be common among all the developers or they may represent an existing application. In any case, there is an actual need to keep the new development separate and in a different directory structure. The Ada/C-C++ Changer allows you to have your source in multiple directories or in multiple libraries. You can use the GUI to point to each directory separately. During the conversion process, the tool will pull the source code from each of the separate directories and pull them all together to create the C/C++ code for the target environment.

You can develop in Ada, but debug in C/C++

A big problem for developers switching between the languages has to do with the build/compile/debug process. In most situations, the developer will make changes to the source code, download the application, and then debug it. But, in most situations, all of this *is using the same language.*

When you are using two different languages, you run into the problem of modifying code and then debugging in a different language. Here's the scenario, let's say that you are writing in Ada, but then you need to test the application on the processor in C/C++. It becomes difficult to manage this constant transition between the two languages.

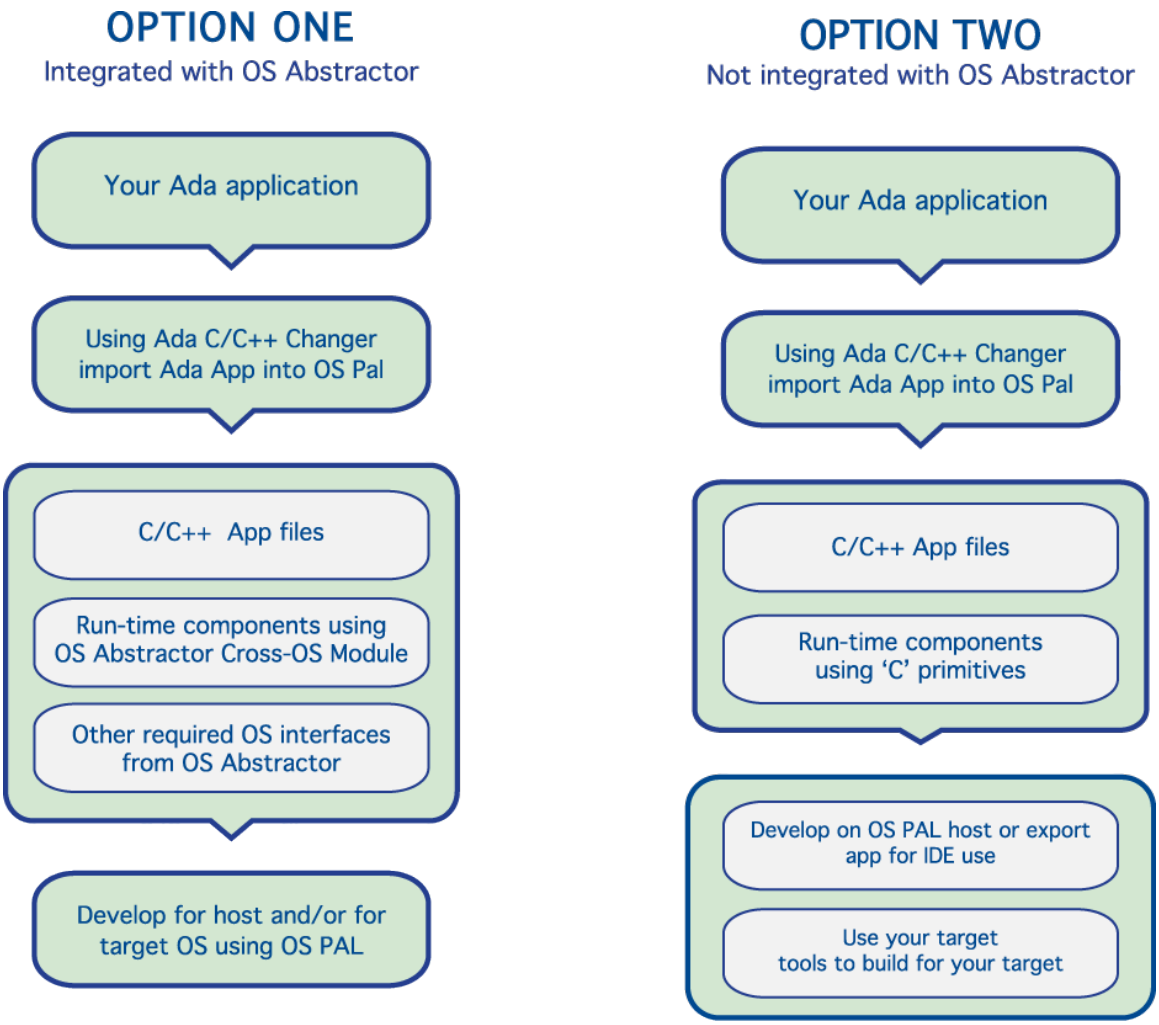
Ada/C-C++ Changer makes this aspect of the project easy. You can develop your code in Ada and then debug in C/C++. By setting certain flags in the GUI, you allow the conversion process to generate a directive in the C/C++ source code that allows most C/C++ debuggers to trace the generated object code back to the particular line of Ada source that produced it. You can also setup the tool so that C/C++ source code is shown in the debugger rather than the Ada source.

Additionally, using Ada/C-C++ Changer you can also see local and global program data during the debugging session and you can set breakpoints in either the Ada source, C /C++source, or disassembly code.

Integrating with a real-time operating system

As you move to your new platform, many applications will want to integrate with real-time operating systems (RTOS). This can be accomplished by using OS Abstractor from MapuSoft. Using OS Abstractor allows you to keep your real-time performance while allowing you the flexibility to move to any platform and real-time operating system you choose. In addition to supporting any real-time operating system you choose, your output code exactly matches the semantics of your Ada application.

An illustration is shown below:



Support for multiple targets, operating systems, and interfaces

Ada-C/C++ Changer can work on targets with no OS or can directly utilize the OS primitives provided by the OS Abstractor to support most of the major operating systems being used in software development today. Below is a list of the supported operating systems:

VxWorks® 6x/5x

Windows® XP/Vista/7/CE

Android®

Linux® 2.4/2.6

LynxOS®

LynxOS-SE®

uITRON®

MQX®

NetBSD®

Nucleus®

QNX® Neutrino® RTOS

RT Linux®

Solaris®

ThreadX®

T-Kernel®

µC/OS-III™

Unix®

FreeRTOS™

For other configurations -

If you need more a complex setup, on the Ada-C/C++ Changer Configuration Options page you can set the following configurations:

C/C++ Output – Allows you to modify the stylistic considerations based on your preferences

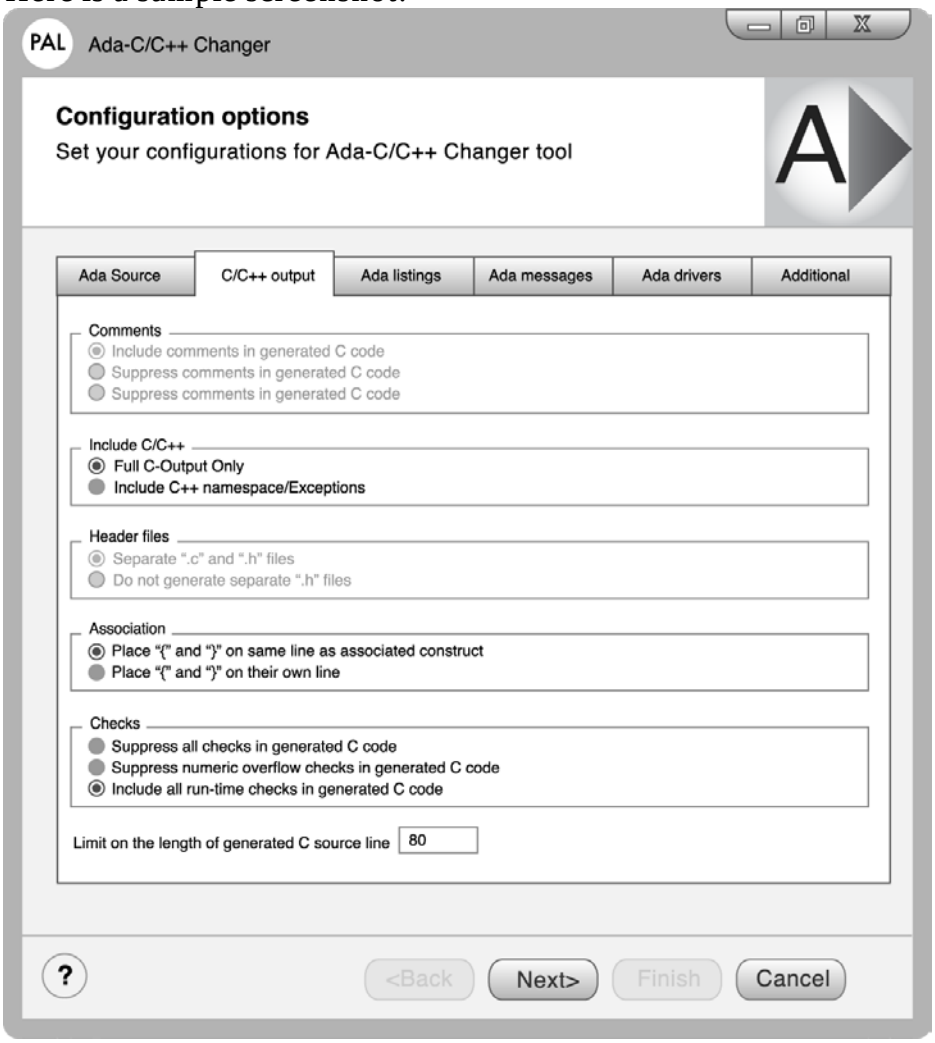
Ada Listings – Allows you to set your preferences for Ada listings

Ada Messages – Controls the output and number of the Ada error and warning messages

Ada Drivers - Specifies the required modes for reporting compiler actions

Additional – Allows you create custom options or additional options

Here is a sample screenshot:



Ada-C/C++ CHANGER®

Conclusion

The Ada-C/C++ Changer is a complete tool for integrating projects that include Ada and C/C++. It allows you to easily convert software written in Ada code to C/C++ utilizing AppCOE. The resultant C/C++ software can be integrated with OS Abstractor® from Mapusoft's Cross-OS Development Platform to support a wide variety of host and target OS platforms. This conversion process eliminates the need for costly and tedious code rewrites, saving you time and money.

The Ada-C/C++ Changer is a fully automatic conversion engine and requires no human intervention. It converts 100% of your Ada source into C/C++. The Changer is based on a fully validated Ada compiler, which supports the full Ada83/Ada 95 language. The output code is efficient and readable C/C++ that exactly matches the semantics of the original Ada program. The Ada-C/C++ Changer also includes a GNAT Ada compatibility option to allow you to easily convert Ada code developed utilizing GNAT compilers. It also includes a POSIX Ada interface package to assist you in transitioning your code that relies on POSIX features.

Ada-C/C++ Changer can work on targets with no OS or can directly utilize the OS primitives provided by the OS Abstractor to support most of the major operating systems being used in software development today.

To get a full demo, please contact us at: info@mapusoft.com or call 1-877-MAPUSOFT (1-877-627-8763).

Ada-C/C++ CHANGER®

About Mapusoft

MapuSoft Technologies (MT) is the number one provider of embedded software re-usability solutions and services that are designed to protect software investment by providing customers a greater level of flexibility and control with product development. In addition to off-the-shelf tools, MT offers porting, integration, support and training services to help developers easily migrate from legacy platforms to the next generation. We believe that our advanced software and vision will revolutionize the embedded software industry. We are working hard to provide software that is practical, familiar, financially reasonable, and easily operable. We provide full source code with no royalty fees. Our licensing strategy makes it extremely affordable for you to incorporate our products into your embedded applications. In addition, our attention to engineering detail provides you with robust software and requires minimal technical maintenance.

About Ada-C/C++ Changer

Ada-C/C++ Changer allows developers to easily convert software written in Ada code to C/C++ utilizing AppCOE. The resultant C/C++ software can be integrated with the robust OS Abstractor® environment to support a wide variety of host and target OS platforms. The automatic conversion process eliminates the need for costly and tedious code rewrites, providing extensive resource savings. Ada-C/C++ Changer generates ANSI C output as well as certain C++ features while preserving Ada code‘s comments, files, structures, and variable names to ease ongoing code maintenance. . The Ada-C/C++ Changer also includes a GNAT Ada compatibility option to allow you to easily convert Ada code developed utilizing GNAT compilers. It also includes a POSIX Ada interface package to assist you in transitioning your code that relies on POSIX features.

For more information

- To download MapuSoft’s free software evaluation visit: http://mapusoft.com/downloads/
- To learn more about our licenses and request a quote visit: http://connect.mapusoft.com/contactus.html
- Ada-C/C++ Changer Datasheet: http://www.mapusoft.com/wp-content/uploads/documents/ada-changer.pdf
- Cross-OS Development Platform Datasheet: http://www.mapusoft.com/wp-content/uploads/documents/cross-os-dev-platform.pdf

Ada-C/C++ CHANGER®

Contact Information

For more information about Ada/C-C++ Changer or Mapusoft, please contact us at:

US Headquarters

MapuSoft Technologies, Inc.
Unit 50197
Mobile, AL 36605

Tel: (251) 665-0280
Toll Free: 1-877-MAPUSOFT (1-877-627-8763)
Fax: (251) 665-0288

www.mapusoft.com

E-mail: info@mapusoft.com or sales@mapusoft.com

For a complete listing of International Offices, please click here.

Mapusoft's AppCOE: