



## Pascal to C++ conversion service

Version 1.2

19 September 2002

Stromasys SA

P.O.Box 156

1228 Plan-les-Ouates

Switzerland

Tel: +41 22 794 1070

Fax: +41 22 794 1073

[www.stromasys.com](http://www.stromasys.com)

This document is provided for information and is not a legally binding offer. Stromasys reserves the right to change service specifications without prior notice or retire the specific service offered. Trademarks are the property of their respective owners.

### *Service availability*

The Pascal to C++ programming language conversion service is available as part of a platform migration project or as a separate service. In the latter case a typical project is split into a fixed-price project evaluation phase and an execution phase.

The fixed price evaluation delivers a detailed project plan within four weeks of its commencement and includes a cost estimate for the conversion. If the project is ordered within two months of completion of the project plan, this estimate can be considered as a fixed-price quotation.

The Pascal to C++ programming language conversion service provided by Stromasys is a risk free way to transform older Pascal applications into modern C++ code, while maintaining the original application functionality. Such conversion can generate substantial cost savings; older tools, development platforms and maintenance skills are no longer required. Modern platforms and programming languages make it easier to attract good software engineers, maintain code and offer a wider choice of powerful development platforms. Programming language conversion can significantly ease the transition to a new system environment.

Stromasys provides a comprehensive service designed to limit the involvement of your own staff to the final acceptance testing after delivery of the translated applications. We can tailor the conversion process to company specific coding standards, and replace middleware and system calls. Several levels of refinement are available, ranging from a rapid conversion for unmodified further use to development of customer specific methods and classes. In all cases, the original functionality is maintained by default.

### *Service components*

The Pascal to C++ programming language conversion service is based on the following main components:

- A set of source code analysis tools to gather statistical information and to detect constructions and extensions that cannot be converted automatically. This information is used to tune the proprietary P2CPP converter and to extend/develop the C++ runtime libraries if necessary. This analysis also covers the system functions of host platform operating system, an essential part to be considered in the migration process.
- The P2CPP converter engine to handle automated volume conversion of the Pascal source code. The P2CPP converter is parameter controlled and manually adjusted based with the information obtained in the first step. (The P2CPP converter is not a sellable product.)
- The Pascal support C++ runtime library.
- Specific extensions to efficiently handle VAX/DEC Pascal and Oregon Pascal.
- Specific emulation libraries for Operating System dependent code segments. Stromasys developed technology templates for migrations between different host platforms (e.g. from VAX VMS/OpenVMS to industry-standard UNIX or Windows 2000 systems).

### *The conversion service includes*

- A feasibility analysis to determine manual and automatic components.
- Pre-conversion code preparation, cleaning up, filtering and processing with 'standardization' tools.
- The analysis of OS-specific extensions and packages used in the application.
- Bulk conversion of Pascal code to C++ using the P2CPP converter.
- Generation of OS-specific migration components for the new platform.
- Post-conversion clean-up (e.g. inline comments)
- Manual conversion of the outstanding conversion issues as reported by the initial analysis and P2CPP.

### Code conversion principles

From a design point of view, Pascal and C++ have radically different philosophies in implementing an application design. Pascal places an emphasis on routines and structural design, while C++ places its emphasis on classes and object orientation.

There are also some specific data types and control statements that cannot be directly converted to C++ statements without introducing emulation. Most of the control statements and data types naturally convert to C++ with no loss in readability, but some compromises for Pascal-specific constructions are made in the converted code.

### Conversion vs. emulation

A predefined Pascal language environment consists of a set of standard functions provided by every Pascal implementation.

Some of the procedures implemented in these packages can be converted to native C++ functions, while others need special emulation provided by the Pascal support C++ runtime library.

### DEC Pascal

The DEC Pascal language environment has additional sets of capabilities for working with RMS files, System Services and VMS Run-Time Libraries.

Some of the procedures and objects implemented in these packages can be converted to native C++ runtime classes and functions, while others require special classes and objects from the VMS Emulation Library (VEL) of Stromasys.

- Converted source code style improvements and redesign (if required).

### Code conversion principles

The following specific constructions and situations are handled separately during Pascal to C++ conversion:

- **Functions and subroutines**  
Pascal functions and subroutines are converted to C++ functions. Nested subroutines are moved out of outer subroutines to the global scope (possibly with prefixes in names to resolve name conflicts). Variables from outer functions that are also used in nested functions are not moved automatically to global scope, but reported by the analysis tools for separate processing.
- **Data types**  
Predefined data types such as *integer* or *real* are translated to the corresponding C++ build-in types via C++ *typedef* statements. This approach allows type redefinition during the software lifecycle.
- **Strings**  
Strings are converted to the *varying\_string* class from the support C++ runtime library. This approach allows to emulate Pascal string layout with the first two bytes for the length and the remainder for the string data itself.
- **Arrays**  
Arrays are converted to the *array* class from the support C++ runtime library. C/C++ native arrays are not used as they do not cover the correct array and conformant array parameters support.
- **SET data type**  
The Pascal SET data type is converted to the class *set* from the support C++ runtime library. Class *set* is implemented very efficiently in terms of performance and memory requirements (only 32 bytes per *set* object).
- **Conformant array parameters**  
Conformant array parameters are translated to the *conf\_array* class from the support C++ runtime library.
- **Variant records**  
Variant records are converted to C++ structures with unions.
- **Schema type**  
Schema types are converted to template classes in the resulting C++ code.
- **Access data types**  
Access data types are converted to standard C++ pointers.

### The major Pascal subroutines groups and solutions used for each group

- **File I/O (open, close, rewrite, put, get, reset, read, write)**  
File handling procedures are emulated by the class *file* from the Pascal support C++ runtime library.
- **Dynamic allocation procedures (new, dispose)**  
These procedures are converted to the native C++ memory management operators *new* and *delete*.
- **Strings**  
Procedures from this group are translated to the appropriate procedures from the support C++ library.
- **Arithmetic functions**  
Procedures from this package are converted to native C++ arithmetic functions.
- **Transfer functions (trunc, round, ord, pred, succ, chr)**  
These functions are implemented as macros in the Pascal support C++ runtime library.
- **Text I/O (writeln, readln)**  
These procedures are converted to C++ streamio operators (<< and >>) of the class *file* implemented in the Pascal support C++ runtime library.