FUJITSU

FUJITSU Software Compiler Package V1.0L20

Overview

J2UL-2578-02ENZ0(00)
July 2020

Preface

Purpose of This Manual

This manual provides an overview of the functions of Compiler Package (hereafter referred to as this software). It also describes compatibility with existing products for PRIMEHPC FX1000.

This software supports PRIMEHPC FX700 system (hereinafter referred to as "FX system").

Intended Readers

This manual is written for users who develop scientific and technical computation programs, and for system administrators who install this software and set the environment settings.

Organization of This Manual

This manual consists of the following chapters:

Chapter 1 Compiler Package Overview

Provides an overview of the software.

Chapter 2 Functions of Components

Provides an overview of the function of each component.

Chapter 3 Compatibility with PRIMEHPC FX1000 system

Describes compatibility with a Technical Computing Suite V4 for PRIMEHPC FX1000 system.

Export Controls

Exportation/release of this manual may require necessary procedures in accordance with the regulations of your resident country and/or US export control laws.

Trademarks

- Linux(R) is the registered trademark of Linus Torvalds in the U.S. and other countries.
- OpenMP is a trademark of OpenMP Architecture Review Board.
- Mac(R) and macOS(R) are registered trademarks of Apple Inc.
- Microsoft, Windows, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- All other trademarks are the property of their respective owners.
- The trademark notice symbol (TM, (R)) is not necessarily added in the system name and the product name, etc. published in this material.

Date of Publication and Version

Version	Manual code
July 2020, 2nd Version	J2UL-2578-02ENZ0(00)
February 2020, 1st Version	J2UL-2578-01ENZ0(00)

Copyright

Update History

Changes	Location	Version
Added versions of OpenMP standards supported by the compilers.	1.1	2nd Version
Added "Release Note" to the manual list.	1.5	
Added versions of standards supported by the Fortran compiler.	2.1.1	
Added versions of standards supported by the C compiler.	2.1.2	
Added versions of standards supported by the C++ compiler.	2.1.3	
Added additional compatibility notes.	Chapter 3	
Changed the look according to product upgrades.	-	

Contents

Chapter 1 Compiler Package Overview

This chapter presents an overview of this software.

1.1 What is Compiler Package?

This software supports development and execution of high-performance parallel programs written in Fortran, C, or C++.

This software has the following features:

Assists with development of "high-functionality" parallel programs

- Compiler optimization technology that elicits high CPU execution performance
- Various functions that assist with highly parallelized programming using hybrid parallels (*)
  - Compiler automatic parallelization function that simplifies thread parallelization
  - MPI communication library that underpins process parallelization
  - Optimized mathematical library and its parallelized edition (thread parallels, MPI parallels)

  *) Hybrid parallel is a parallel programming model containing both thread parallels and process parallels.
- Optimization support using FX system specific functions (HPC Extension Function)

Assists with "efficient" development of large-scale parallel programs

- Compilers equipped with high-level optimization functions and automatic parallelization functions
- Profiler that underpins program performance tuning
- Optimized mathematical library and its parallelized edition

Assists with development of "highly portable" programs

- Compilers that conform to international program language standards (Fortran, C, and C++)
- Support of various industry standard specifications
  - C and C++ compilers that support GNU compiler extended specifications
  - Compilers that conform to OpenMP 4.0, OpenMP 4.5 and a subset of OpenMP 5.0
  - MPI library which conforms to MPI-3.1 standard and a subset of MPI-4.0 standard (tentative name)
  - Provision of BLAS, LAPACK, and ScaLAPACK

1.2 Applicable Fields

This software is effective in developing high performance technical applications and exert most capability in the following:

- Development and execution of scientific and technical computing application programs written in Fortran, C, and C++
- Development and execution of highly parallelized application programs for process parallels and thread parallels

1.3 System on which this Software Runs

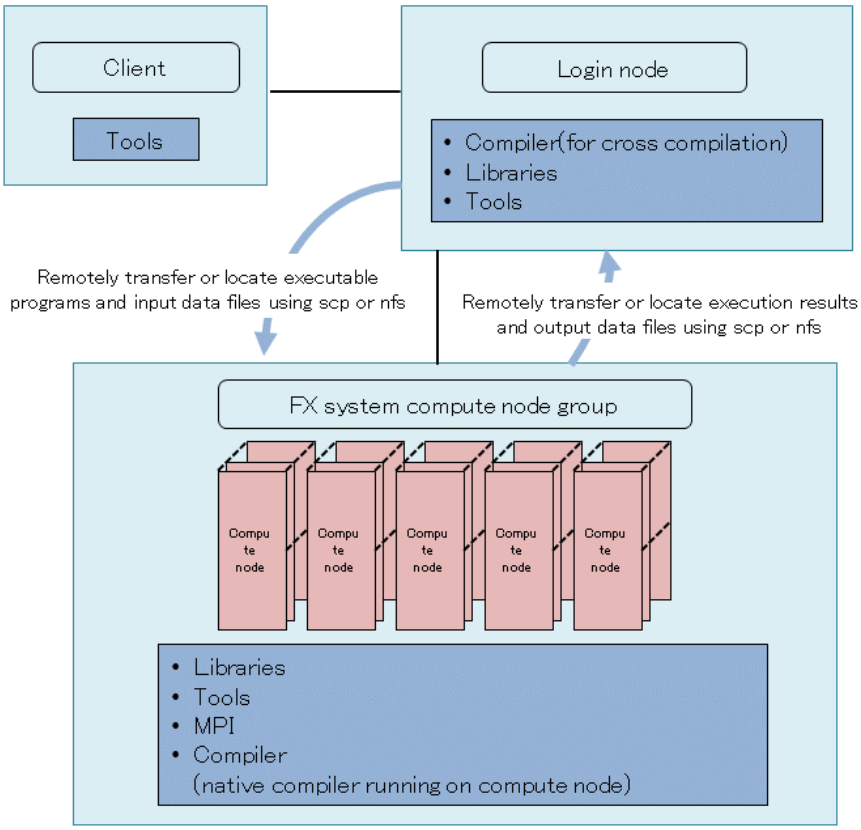
The types of system running this software can be broadly classified as follows:

1. Client
2. Login node
3. FX system (compute node group)

FX system is always required to use this software, but login node (PRIMERGY system) and client are optional. Login node (PRIMERGY system) is required in order to use the cross-compilation function of this software or the profile result output command. Client is required to view the profile result as CPU Performance Analysis Report.

The example of system constitution is shown below.

Figure 1.1 Example of System constitution



1.3.1 Client (Windows Operating System/ Mac Operating System/ Linux Operating System)

Client is the computer that displays the profiling results as CPU Performance Analysis Report.

- Tools
  - CPU Performance Analysis Report

1.3.2 Login Node (Linux Operating System)

Login Node is the node on which the various compilers (for cross compilation) for FX system can run. End users can create an executable program for FX system by cross-compilation and link-editing.

Specifically, install the following components on the login node:

- Compiler (for cross compilation)
  - Fortran compiler
  - C compiler
  - C++ compiler

    - MPI program compilation commands
- Libraries
    - Runtime libraries of each compiler
    - Mathematical library
    - MPI library
    - Library used for Software development support tools
- Tools
    - Processing part on the login node side of the Software development support tools
    - Commands to output the Profiling Results

1.3.3 FX System (Compute Nodes)

Compute nodes of FX system are nodes that run an executable program for an FX system.

There are also compilers (for native compilation) that run on each compute node of FX system, which allow you to compile and link-edit programs on the compute node.

Specifically, install the following components on each compute node:

- Libraries
    - Runtime libraries of each compiler
    - Mathematical library
    - MPI library
    - Libraries, which are linked to executable files, of software development support tools
- Tools
    - Commands to get Profiling Data
- MPI
    - A command to execute MPI executable programs
- Compiler (for native compilation)
    - Fortran compiler
    - C compiler
    - C++ compiler
    - MPI program compilation commands
- HPC Extension Function
    - HPC tag address override control function (control of processor-specific function of FX system)
    - Inter-core hardware barrier driver/library
    - Sector cache driver/library
    - Large page library

1.4 Program Development Flow

This section describes the flow of program development (Program creation, compilation, execution, debugging, tuning) using a file system (hereafter referred to as shared file system) that is shared by login nodes and compute nodes.

1.4.1 Creating and Executing a Program

1. Creating a program

   Create a source program on the login node, using various functions provided by the Mathematical library and MPI library as needed.

2. Compiling the program

   Create an executable program by compiling the source program with the cross-compilers on the login node, and place it on the shared file system.

   For instructions on how to use the compilers, read the user's guide for each compiler. Read "MPI User's Guide" for instructions on compiling MPI programs.

3. Executing a program

   Since the executable program created on the login node is also shared on the compute node, execute the executable program on compute nodes.

   Read "MPI User's Guide" for more information on running MPI programs.

4. Checking the results

   Check the execution results on the login node or the compute nodes.

1.4.2 Debugging a Program

This software does not provide any source-level debugger. Use gdb for source-level debugging.

1. Compiling the program

   On the login node, compile the source program to create an executable program for debugging, with the corresponding compiler. Specify the compiler option -g when compiling the program.

2. Debugging with gdb

   Run the executable program created on the login node on the compute node and debug using gdb. Specifically, debugging is performed by executing an executable program under gdb, or by attaching gdb to the executing program.

1.4.3 Tuning a Program

Refer to the "Profiler User's Guide" for tuning details.

1. Compiling the program

   On the login node, create an executable program by compiling the source program.

2. Performance measurement

   Since the executable program created on the login node is also file shared on the compute node, execute the executable program on compute nodes to measure profile data. In order to measure the profile data, execute the executable program through the profiler command ("fipp" or "fapp").

3. Output profiling results

   On the login node, use the fipppx or fapppx command to output profiling results from profiling data.

4. Check profile results

   On the login node, consider where to tune from the profile results.

5. Editing the program (Modifying the program for tuning)

   Modify and improve the source program.

1.5 Manual List

The following manuals are provided with this software:

Table 1.1 Manual List

Manual Name	Contents	Format
Overview	An overview of Compiler Package (this document)	PDF
Release Note	Describes new features supported by this software and compatibility with the previous version	PDF
Fortran Language Reference	Describes the Fortran language specification supported by the Fortran compiler.	PDF
Fortran User's Guide	How to use the Fortran compiler How to program in Fortran	PDF
Fortran User's Guide Additional Volume COARRAY	How to use the COARRAY feature How to programming with COARRAY feature	PDF
C User's Guide	How to use the C compiler Description of extended language specifications based on the C and OpenMP specifications for the C compiler	PDF
C++ User's Guide	How to use the C++ compiler Description of extended language specifications based on the C++ and OpenMP specifications for the C++ compiler	PDF
Fortran Compiler Messages	Descriptions of messages during compilation using the Fortran compiler	PDF
C/C++ Compiler Optimization Messages	Descriptions of optimization messages from the C and C++ compilers	PDF
Fortran/C/C++ Runtime Messages	Description of runtime messages from programs written in Fortran, C, and C++	PDF
MPI User's Guide	How to use the MPI library	PDF
MPI User's Guide Additional Volume Java Interface	MPI Library Java Interface Description	HTML
Profiler User's Guide	Description of features and usage of the profiler, a performance analysis tool	PDF
Programmer's Guide for Usage of Mathematical Libraries	How to use the Mathematical Libraries	PDF
SSL II User's Guide	Descriptions of the functions and use of the Scientific Subroutine Library II (SSL II)	PDF
FUJITSU SSL II Extended Capabilities User's Guide		PDF
FUJITSU SSL II Extended Capabilities User's Guide II		PDF
Fujitsu SSL II Thread-Parallel Capabilities User's Guide	Descriptions of the functions and usage of the Scientific Subroutine Library II Thread-Parallel Capabilities	PDF
Fujitsu C-SSL II User's Guide	How to use C-SSLII (C-Scientific Subroutine Library II), a scientific library available from C	PDF
Fujitsu C-SSL II Thread-Parallel Capabilities User's Guide	How to use the functions and usage of the C Scientific Subroutine Library II Thread-Parallel Capabilities	PDF
Fujitsu SSL II/MPI User's Guide	Descriptions of the functions and usage of the Scientific Subroutine Library II/MPI (SSL II/MPI)	PDF
BLAS LAPACK ScaLAPACK User's Guide	Obtain an overview of the functions provided by BLAS, LAPACK and ScaLAPACK, and how they relate to the BLAS, LAPACK, and ScaLAPACK published by Netlib	PDF

Manual Name	Contents	Format
Fast Basic Operations Library for Quadruple Precision User's Guide	How to use Fast Basic Operations Library for Quadruple Precision available in Fortran and C++	PDF

View PDF manuals in the latest Adobe(R) Reader(R) version.

Chapter 2 Functions of Components

2.1 Compiler

2.1.1 Fortran Compiler

The Fortran compiler compiles programs written in Fortran and can create executable programs that are highly optimized, in particular to achieve high CPU execution performance for targeted compute nodes. If required, the Fortran compiler can also create executable programs capable of automatic parallelization or thread-level parallel execution using OpenMP specifications. An execution environment for the fast and secure execution of these executable programs is also prepared.

An overview of Fortran compiler functions is presented below. Refer to the "Fortran User's Guide" for Fortran compiler details.

1. Language specifications

   The Fortran compiler supports the following standards:

   - ISO/IEC 1539-1:2018 (Fortran 2018 standard) a subset of specification
   - ISO/IEC 1539-1:2010 (Fortran 2008 standard)
   - ISO/IEC 1539-1:2004, JIS X 3001-1:2009 (Fortran 2003 standard)
   - ISO/IEC 1539-1:1997, JIS X 3001-1:1998 (Fortran 95 standard)
   - Fortran 90 standard and FORTRAN 77 standard
   - OpenMP API Version 5.0 a subset of specification
   - OpenMP API Version 4.5
   - OpenMP API Version 4.0

   The Fortran compiler also supports the main industry standard Fortran language specifications. Refer to the "Fortran Language Reference" for information on the language specifications supported by the Fortran compiler.

2. Main functions

   The Fortran compiler is equipped with an "Automatic parallelization function" that enables programs that automatically perform thread-level parallel processing to be created just by specifying the compile options. The Fortran compiler also supports the "OpenMP API specifications" that perform thread-level parallel processing by means of directive lines specified within the program. A shared memory system is a prerequisite for the automatic parallelization and parallel processing that use OpenMP specifications, and these functions are enabled within the compute nodes. Used in conjunction with the MPI library, these functions support the highly efficient hybrid parallel programming model (thread parallels + MPI process parallels).

   The Fortran compiler also supports a wide variety of other functions, such as program debugging (procedure reference verification, undefined data reference, array size validity), precision sensitivity diagnosis through application of precision reduction, trace-back map, error monitoring, and output of compilation information containing parallelization and optimization information.

3. Optimization functions

   The Fortran compiler is equipped with optimization functions listed below, and can create object programs capable of fast execution at compute nodes. In addition, various optimization control lines are provided in order to facilitate these optimizations from within programs.

   - Optimizations reconfiguring nested loops
   - Instruction scheduling suitable for the CPU characteristics
   - Increasing degree of parallelism by SIMD utilizing SVE
   - Reducing save and restore instruction counts for register contents
   - Efficient use of cache by means of prefetch instructions

- Optimization functions that effectively use the HPC tag address override function of A64FX processor:
  - Control of hardware prefetch and software prefetch
  - Software control of sector cache by optimization control lines

2.1.2 C Compiler

The C compiler compiles programs written in C and, like the Fortran compiler, can create executable programs highly optimized to achieve high CPU execution performance for targeted compute nodes. If required, the C compiler can also create executable programs capable of automatic parallelization or thread-level parallel execution using OpenMP specifications.

An overview of C compiler functions is presented below. Refer to the "C User's Guide" for C compiler details.

1. Language specifications

   The C compiler supports the following standards:

   - ISO/IEC 9899:2011 (C11 standard)
   - ISO/IEC 9899:1999 (C99 standard)
   - ISO/IEC 9899:1990 (C89 standard)
   - OpenMP API Version 5.0 a subset of specification
   - OpenMP API Version 4.5
   - OpenMP API Version 4.0

   The C compiler also supports GNU compiler extended specifications.

2. Main functions

   The C compiler is equipped with an "Automatic parallelization function" that enables programs that automatically perform thread-level parallel processing to be created just by specifying the compile options. The C compiler also supports the "OpenMP API specifications" that perform thread-level parallel processing by means of directive lines specified within the program. A shared memory system is a prerequisite for the automatic parallelization and parallel processing that use OpenMP specifications, and these functions are enabled within the compute nodes. Used in conjunction with the MPI library, these functions support the highly efficient hybrid parallel programming model (thread parallels + MPI process parallels).

   The C compiler also supports GNU compiler extended specifications and has superior portability from a variety of systems.

3. Optimization functions

   The C compiler is equipped with optimization functions listed below, and can create object programs capable of fast execution at compute nodes. In addition, various optimization control lines are provided in order to facilitate these optimizations from within programs.

   - Optimizations reconfiguring nested loops
   - Instruction scheduling suitable for the CPU characteristics
   - Increasing degree of parallelism by SIMD utilizing SVE
   - Reducing save and restore instruction counts for register contents
   - Efficient use of cache by means of prefetch instructions
   - Optimization functions that effectively use the HPC tag address override function of A64FX processor:
     - Control of hardware prefetch and software prefetch
     - Software control of sector cache by optimization control lines

2.1.3 C++ Compiler

The C++ compiler compiles programs written in C++ language and, like the Fortran compiler, can create executable programs highly optimized to achieve high CPU execution performance for targeted compute nodes. If required, the C++ compiler can also create executable programs capable of automatic parallelization or thread-level parallel execution using OpenMP specifications.

An overview of C++ compiler functions is presented below. Refer to the "C++ User's Guide" for C++ compiler details.

1. Language specifications

   The C++ compiler supports the following standards:

   - ISO/IEC 14882:2017 (C++17 standard)
   - ISO/IEC 14882:2014 (C++14 standard)
   - ISO/IEC 14882:2011 (C++11 standard)
   - ISO/IEC 14882:2003 (C++03 standard)
   - OpenMP API Version 5.0 a subset of specification
   - OpenMP API Version 4.5
   - OpenMP API Version 4.0

   The C++ compiler also supports GNU compiler extended specifications.

2. Main functions

   The C++ compiler is equipped with an "Automatic parallelization function" that enables programs that automatically perform thread-level parallel processing to be created just by specifying the compile options. The C++ compiler also supports the "OpenMP API specifications" that perform thread-level parallel processing by means of directive lines specified within the program. A shared memory system is a prerequisite for the automatic parallelization and parallel processing that use OpenMP specifications, and these functions are enabled within the compute nodes. Used in conjunction with the MPI library, these functions support the highly efficient hybrid parallel programming model (thread parallels + MPI process parallels).

   The C++ compiler also supports GNU compiler extended specifications and has superior portability from a variety of systems.

3. Optimization functions

   The C++ compiler is equipped with optimization functions listed below, and can create object programs capable of fast execution at compute nodes. In addition, various optimization control lines are provided in order to facilitate these optimizations from within programs.

   - Optimizations reconfiguring nested loops
   - Instruction scheduling suitable for the CPU characteristics
   - Increasing degree of parallelism by SIMD utilizing SVE
   - Reducing save and restore instruction counts for register contents
   - Efficient use of cache by means of prefetch instructions
   - Optimization functions that effectively use the HPC tag address override function of A64FX processor:
     - Control of hardware prefetch and software prefetch
     - Software control of sector cache by optimization control lines

2.2 Communication Library

2.2.1 MPI Library

The MPI library (message passing library) conforms to the MPI-3.1 standard and a subset of the MPI-4.0 standard (tentative name) prescribed by the MPI Forum. The MPI library supports the interconnect, known as InfiniBand.

The MPI library can be used from Fortran compiler, C compiler, C++ compiler, and Java.

Refer to the "MPI User's Guide" for information on the MPI library.

2.3 Software Development Support Tools

2.3.1 Profiler

The profiler is a performance analysis tool that can measure various types of information required for analyzing the performance of application programs written in Fortran, C, and C++. The profiler can also measure profiler information for programs that support thread parallels and MPI process parallels. The profiler is comprised of Instant Performance Profiler, Advanced Performance Profiler, and CPU Performance Analysis Report.

- Instant Performance Profiler

  Instant Performance Profiler helps users understand performance tendency of an application without recompilation. It is possible to measure performance information with low overhead for large-scale parallel programs.

  The output items of Instant Performance Profiler are time statistics information, CPU activity information, cost information, call graph information, and source code information.

- Advanced Performance Profiler

  Advanced Performance Profiler helps users understand the detailed performance of specific regions.

  The output items of Advanced Performance Profiler are time statistics information, MPI communication cost information, and CPU performance analysis information.

- CPU Performance Analysis Report

  CPU Performance Analysis Report aggregates CPU performance analysis information measured by Advanced Performance Profiler, and visualizes them using tables and graphs so that users can easily understand them.

Refer to the "Profiler User's Guide" for profiler details.

2.4 Mathematical Libraries

In addition to Fujitsu's own mathematical libraries (SSL II and C-SSL II) which are widely used within Japan by R&D users, the linear algebra field libraries (BLAS, LAPACK, and ScaLAPACK) developed in the US are also provided. Fast Basic Operations Library for Quadruple Precision, which is a library in which a quadruple precision number is expressed in double-double format and arithmetic operations are performed on such formatted numbers, is also provided.

These mathematical libraries (except for ScaLAPACK) make it possible for the same routine to be called from multiple threads simultaneously (thread safe).

Important SSL II functions also provide the thread parallel routines that describe the parallel mathematical computing algorithms intended for shared memory type scalar parallel computing in OpenMP Fortran. In addition, these parallel mathematical computing algorithms are also provided as C-SSL II thread parallel routines.

Three-dimensional Fourier transforms parallelized by MPI are provided by SSL II/MPI.

Thread parallel routines are also provided for the major BLAS routines and for the major LAPACK routines.

All of these mathematical libraries are tuned so that optimized execution performance is obtained at each compute node.

Refer to the "Programmer's Guide for Usage of Mathematical Libraries" for information on Usage of Mathematical Libraries.

Refer to the "SSL II User's Guide", "FUJITSU SSL II Extended Capabilities User's Guide", "FUJITSU SSL II Extended Capabilities User's Guide II", and "Fujitsu SSL II Thread-Parallel Capabilities User's Guide" for information on SSL II.

Refer to the "Fujitsu C-SSL II User's Guide" and the "Fujitsu C-SSL II Thread-Parallel Capabilities User's Guide" for information on C-SSL II.

Refer to the "Fujitsu SSL II/MPI User's Guide" for information on SSL II/MPI.

Refer to the "BLAS LAPACK ScaLAPACK User's Guide" for information on BLAS, LAPACK and ScaLAPACK.

Refer to the "Fast Basic Operations Library for Quadruple Precision User's Guide" for information on Fast Basic Operations Library for Quadruple Precision.

2.5 HPC Extension Function

The HPC extension function provides various extended drivers/libraries for the FX system to use standard Linux functions.

HPC tag address override control function

This function controls the processor-specific HPC tag address override function of the FX system.

Inter-core hardware barrier driver/library

This driver/library provides a driver and library to support the inter-core hardware barrier function on the FX system.

The inter-core hardware barrier function is a function for high-speed synchronization between the threads of a thread-parallelized application program. This function can be used by applications created with compilers included in this software. For details, read the user's guide for each compiler.

Sector cache driver/library

This driver/library provides a driver and library to support the sector cache function on the FX system.

The sector cache function is a function that improves the operating speed of applications by constantly keeping as much as possible highly reusable data in cache. This function can be used by applications created with compilers of this software. For details, read the user's guide for each compiler.

Large page library

The FX system uses Huge Pages, a standard feature in Linux. The two types of huge pages in Linux are THP (Transparent Huge Page) and HugeTLBfs. The FX server adopts the use of HugeTLBfs due to its memory usage efficiency, the range of memory area covered

by enabled Huge Pages, guaranteed acquisition of huge pages, etc.

The large page library provided by the HPC extension function extends the functionality for the FX system so that HugeTLBfs can be used more efficiently and also with higher extensibility. Furthermore, a tool is provided to collect the status of Huge Pages memory usage.

Users can create applications using huge pages by linking this library in compilers of this software. For details, see the user's guide for each compiler.

Chapter 3 Compatibility with PRIMEHPC FX1000 system

This chapter provides notes on compatibility with Development Studio of Technical Computing Language V4 for PRIMEHPC FX1000 system.

Fortran, C, and C++ programs

Basically, source code, object programs, and executables are all compatible(*1). However, an object or executable program might not be compatible under the following conditions:

- There are incompatibilities(*2) due to different product versions of compilers
- There are incompatibilities due to different operating system versions
- MPI programs using system-specific libraries are included in the user program

In such cases, please modify the source program as necessary and recompile it using this software.

*1) This software compatibility is primarily for Technical Computing Suite V4.0L20.

*2) There are some incompatibilities between the compilers and libraries of Technical Computing Suite V4.0L20 and the prior version, Technical Computing Suite V4.0L10. Please note that programs written in Technical Computing Suite V4.0L10 may not be available out-of-the-box when used in conjunction with this software. For more information on these incompatibilities and possible solutions, read "Release Note" of Technical Computing Suite V4.0L20 Development Studio.

Other

In addition, there have been some usage changes for component functions provided by this software. For example, changes have been made to the way threads bind to CPUs, inter-core hardware barriers, and sector cache are used when executing OpenMP programs. For more information, read the user's guide of each component.