インテル® C++ Composer XE 2011 Windows* 版インストール・ガイドおよびリリースノート

資料番号: 321414-003JA
2010 年 11 月 8 日

目次

1 概要

このドキュメントでは、製品のインストール方法、新機能、変更された機能、注意事項、および製品ドキュメントに記述されていない既知の問題について説明します。

インテル® C++ Composer XE 2011 は、以前「インテル® C++ コンパイラー・プロフェッショナル・エディション」と呼ばれていた製品の最新バージョンです。

1.1 変更履歴

このセクションでは製品アップデートにおける重要な変更内容を説明します。

これは最初の製品リリースです。

1.2 製品の内容

インテル® C++ Composer XE 2011 Windows 版*には、次のコンポーネントが含まれています。

- インテル® C++ コンパイラー XE 12.0。Windows* オペレーティング・システムを実行する IA-32 およびインテル® 64 アーキテクチャー・システムで動作するアプリケーションをビルドします。
- インテル® インテグレーテッド・パフォーマンス・プリミティブ 7.0 Update 1
- インテル® マス・カーネル・ライブラリー 10.3
- インテル® スレッディング・ビルディング・ブロック 3.0 Update 3
- インテル® Parallel Debugger Extension
- Microsoft* 開発環境への統合
- サンプルプログラム
- 各種ドキュメント

1.3 動作環境

アーキテクチャー名についての説明は、http://software.intel.com/en-us/articles/intel-architecture-platform-terminology/ (英語) を参照してください。

- インテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) 対応の IA-32 またはインテル® 64 アーキテクチャー・プロセッサーをベースとするコンピューター (インテル® Pentium® 4 プロセッサー以降、または互換性のあるインテル以外のプロセッサー)
    - 機能を最大限に活用できるよう、マルチコアまたはマルチプロセッサー・システムの使用を推奨します。
- RAM 1GB (2GB 推奨)

- 4GB のディスク空き容量 (すべての機能およびすべてのアーキテクチャー)
- Microsoft* Windows* XP、Microsoft* Windows Vista*、Microsoft* Windows* 7、Microsoft* Windows Server* 2003、Microsoft* Windows Server* 2008、Microsoft* Windows HPC Server* 2008 (エンベデッド・エディションはサポートされていません)
    - Microsoft* Windows Server* 2008 または Windows HPC Server* 2008 では Microsoft* Visual Studio* 2008 SP1 が必要です。下記にリストされている Visual Studio* のその他のバージョンは Windows Server* 2008 または Windows HPC Server* 2008 ではサポートされていません。
- IA-32 対応アプリケーションまたはインテル® 64 対応アプリケーションのビルドに、Microsoft* Visual Studio* 開発環境あるいはコマンドライン・ツールを使用する場合は、次のいずれか:
    - Microsoft* Visual Studio* 2010 (C++ コンポーネントと [X64 コンパイラおよびツール] コンポーネントがインストールされていること) [1]
    - Microsoft* Visual Studio* 2008 Standard Edition 以上 (C++ と [x64 コンパイラおよびツール] コンポーネントがインストールされていること) [1]
    - Microsoft* Visual Studio* 2005 Standard Edition 以上 (C++ と [x64 コンパイラおよびツール] コンポーネントがインストールされていること) [1]
- IA-32 アーキテクチャー・アプリケーションのビルドに、コマンドライン・ツールのみを使用する場合は、次のいずれか:
    - Microsoft* Visual C++* 2010 Express Edition
    - Microsoft* Visual C++* 2008 Express Edition
    - Microsoft* Visual C++* 2005 Express Edition と Windows Server* 2008 および .NET Framework 3.5 用 Microsoft* Windows SDK
- インテル® 64 対応アプリケーションのビルドのみにコマンドライン・ツールを使用する場合は、次のいずれか:
    - Windows* 7 用 Microsoft* Windows Software Development Kit Update
    - Windows Server* 2008 および .NET Framework 3.5 用 Microsoft* Windows SDK
- ドキュメントの参照用に Adobe* Reader* 7.0 以降

説明:

1. Microsoft* Visual Studio* 2005/2008 Standard Edition では、[x64 コンパイラおよびツール] コンポーネントがデフォルトでインストールされます。Professional 以上のエディションでは、[カスタム] インストールが必要です。Microsoft* Visual Studio* 2010 では、このコンポーネントがデフォルトで含まれています。
2. インテル® コンパイラーは、デフォルトで、インテル® SSE2 命令対応のプロセッサー (例: インテル® Pentium® 4 プロセッサー) が必要な IA-32 アーキテクチャー・アプリケーションをビルドします。コンパイラー・オプションを使用して任意の IA-32 アーキテクチャー・プロセッサー上で動作するコードを生成できます。ただし、アプリケーションでインテル® インテグレーテッド・パフォーマンス・プリミティブまたはインテル® スレッディング・ビルディング・ブロックを使用している場合、そのアプリケーションの実行には、インテル® SSE2 命令対応のプロセッサーが必要です。
3. アプリケーションは、上記の開発用と同じ Windows* バージョンで実行できます。また、Windows* XP よりも前の非エンベデッドの Microsoft* Windows* 32 ビット・バージョンでも実行できますが、インテルではこれらの互換性テストは行われていません。開発アプリケーションが、古いバージョンの Windows* にはない Win32* API ルーチンを使用

している可能性があります。アプリケーションの互換性テストをご自身の責任で行ってください。アプリケーションを実行するには、特定のランタイム DLL をターゲットシステムにコピーしなければならないことがあります。

1.3.1 IA-64 アーキテクチャー (インテル® Itanium®) 開発の未サポート

本バージョンでは、IA-64 アーキテクチャー (インテル® Itanium®) システム上、または IA-64 アーキテクチャー・システム向けの開発をサポートしていません。インテル® コンパイラー 11.1 ではまだサポートされています。

1.4 ドキュメント

製品ドキュメントは、「インストール先フォルダー」で示されているように、`Documentation` フォルダーに保存されています。

最適化に関する注意事項

インテル® コンパイラー、関連ライブラリーおよび関連開発ツールには、インテル製マイクロプロセッサーおよび互換マイクロプロセッサーで利用可能な命令セット (SIMD 命令セットなど) 向けの最適化オプションが含まれているか、あるいはオプションを利用している可能性がありますが、両者では結果が異なります。また、インテル® コンパイラー用の特定のコンパイラー・オプション (インテル® マイクロアーキテクチャーに非固有のオプションを含む) は、インテル製マイクロプロセッサー向けに予約されています。これらのコンパイラー・オプションと関連する命令セットおよび特定のマイクロプロセッサーの詳細は、『インテル® コンパイラー・ユーザー・リファレンス・ガイド』の「コンパイラー・オプション」を参照してください。インテル® コンパイラー製品のライブラリー・ルーチンの多くは、互換マイクロプロセッサーよりもインテル製マイクロプロセッサーでより高度に最適化されます。インテル® コンパイラー製品のコンパイラーとライブラリーは、選択されたオプション、コード、およびその他の要因に基づいてインテル製マイクロプロセッサーおよび互換マイクロプロセッサー向けに最適化されますが、インテル製マイクロプロセッサーにおいてより優れたパフォーマンスが得られる傾向にあります。

インテル® コンパイラー、関連ライブラリーおよび関連開発ツールは、互換マイクロプロセッサー向けには、インテル製マイクロプロセッサー向けと同等レベルの最適化が行われない可能性があります。これには、インテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2)、インテル® ストリーミング SIMD 拡張命令 3 (インテル® SSE3)、ストリーミング SIMD 拡張命令 3 補足命令 (インテル® SSSE3) 命令セットに関連する最適化およびその他の最適化が含まれます。インテルでは、インテル製ではないマイクロプロセッサーに対して、最適化の提供、機能、効果を保証していません。本製品のマイクロプロセッサー固有の最適化は、インテル製マイクロプロセッサーでの使用を目的としています。

インテルでは、インテル® コンパイラーおよびライブラリーがインテル製マイクロプロセッサーおよび互換マイクロプロセッサーにおいて、優れたパフォーマンスを引き出すのに役立つ選択肢であると信じておりますが、お客様の要件に最適なコンパイラーを選択いただくよう、他のコンパイラーの評価を行うことを推奨しています。インテルでは、あらゆるコンパイラーやライブラリーで優れたパフォーマンスが引き出され、お客様のビジネスの成功のお役に立ちたい

と願っております。お気づきの点がございましたら、お知らせください。

改訂 #20101101

1.5 サンプル

製品コンポーネントのサンプルは、「インストール先フォルダー」の説明にある Samples フォルダーに用意されています。

1.6 日本語サポート

日本語サポートは最初の製品リリースには含まれていません。

インテル® コンパイラーは、日本語ユーザー向けのサポートを提供しています。エラーメッセージ、ビジュアル開発環境ダイアログ、ドキュメントの一部が英語のほかに日本語でも提供されています。エラーメッセージやダイアログの言語は、システムの言語設定に依存します。日本語版ドキュメントは、Documentation および Samples ディレクトリー以下の ja_JP サブディレクトリーにあります。

日本語サポート版を英語のオペレーティング・システムで使用する場合や日本語のオペレーティング・システムで英語サポート版を使用する場合は、http://software.intel.com/en-us/articles/changing-language-setting-to-see-english-on-a-japanese-os-environment-or-vice-versa-on-windows/ (英語) の説明を参照してください。

1.7 テクニカルサポート

インストール時にコンパイラーの登録を行わなかった場合は、インテル® ソフトウェア開発製品レジストレーション・センターで登録してください。登録を行うことで、サポートサービス期間中 (通常は 1 年間)、製品アップデートと新しいバージョンの入手を含む無償テクニカルサポートが提供されます。

テクニカルサポート、製品のアップデート、ユーザーフォーラム、FAQ、ヒント、およびその他のサポート情報は、http://www.intel.com/software/products/support/ (英語) を参照してください。

注: 代理店がテクニカルサポートを提供している場合は、インテルではなく代理店にお問い合わせください。

2 インストール

2.1 インストール前の準備

2.1.1 64 ビット・アプリケーション用の Visual Studio* の設定

Microsoft* Visual Studio* 2005 または 2008 を使用し、64 ビット・アプリケーション (インテル® 64 アーキテクチャー向け) を開発する場合は、Visual Studio* の構成を変更して、64 ビット・サポートを追加します。

Visual Studio* 2005/2008 Standard Edition または Visual Studio* 2010 Professional Edition 以上を使用する場合は、インテル® 64 対応アプリケーションのビルド用に構成を変更する必要はありません。その他のエディションの場合は、次の操作を行ってください。

1. [コントロール パネル] の [プログラムの追加と削除] から [Microsoft Visual Studio 2005 (または 2008)] を選択し、[変更と削除] をクリックします。[Visual Studio メンテナンス モード] ウィンドウが表示されます。[次へ] をクリックします。
2. [機能の追加と削除] をクリックします。
3. [選択した機能をインストールします] で [言語ツール] の [Visual C++] を展開します。
4. [x64 コンパイラおよびツール] ボックスがオンになっていない場合は、オンにし、[更新] をクリックします。ボックスがオンの場合は、[キャンセル] をクリックします。

Visual C++* Express Edition では 64 ビットの開発はサポートされていません。

2.1.2 Microsoft* Windows Vista* および Microsoft* Windows* 7 でのインストール

Microsoft* Visual Studio* 2005 ユーザーは、*Visual Studio* 2005 Service Pack 1* (VS 2005 SP1) と Visual Studio* 2005 Service Pack 1 Update for Windows Vista* (VS 2005 SP1 ページからリンクが提供) をインストールしてください。これらのアップデートをインストールした後に、管理者権限で Visual Studio* が実行できることを確認してください。実行できない場合、インテル® コンパイラーを使用できません。詳細は、Microsoft* の「Visual Studio* on Windows Vista*」ページ (http://msdn2.microsoft.com/en-us/vstudio/aa948853.aspx (英語)) および関連ドキュメントを参照してください。

2.2 インストール

本製品のインストールには、有効なライセンスファイルまたはシリアル番号が必要です。本製品を評価する場合には、インストール時に [製品を評価する (シリアル番号不要)] オプションを選択してください。

DVD で製品を受け取った場合、製品 DVD を DVD ドライブに挿入します。自動でインストールが開始されます。自動で開始されない場合は、Windows* エクスプローラで DVD ドライブのトップレベル・ディレクトリーを開き、setup.exe をダブルクリックします。

製品のダウンロード版を購入した場合は、ダウンロードしたファイル (.EXE) をダブルクリックして、インストールを開始します。利用可能なダウンロード・ファイルには各種あり、それぞれ異なるコンポーネントの組み合わせを提供していることに注意してください。ダウンロード・ページを注意深くお読みになり、適切なファイルを選択してください。

新しいバージョンをインストールする前に古いバージョンをアンインストールする必要はありません。新しいバージョンは古いバージョンと共存可能です。

2.2.1 インテルのアクティベーション・ツールを使用した製品のアクティベーション

この製品リリースでは、新しいインテルのアクティベーション・ツール “ActivationTool.exe” が “[Common Files]\Intel\Parallel Studio XE 2011\Activation\” にインストールされます。

インストール中に評価用ライセンスまたは評価用シリアル番号を使用したり、あるいは [製品を評価する (シリアル番号不要)] オプションを選択して製品をインストールした場合、製品を購入した後にこのアクティベーション・ツール ([スタート] - [すべてのプログラム] -[インテル(R) Parallel Studio XE 2011] -[Product Activation (製品のアクティベーション)]) を使用して製品をアクティベートできます。これにより、評価版から製品版へ移行することができます。

2.2.2 ライセンスサーバーの使用

「フローティング・ライセンス」を購入された場合は、ライセンスファイルまたはライセンスサーバーを使用したインストール方法について http://software.intel.com/en-us/articles/licensing-setting-up-the-client-floating-license/ (英語) を参照してください。この記事には、多様なシステムにインストールすることができるインテル・ライセンス・サーバーに関する情報も記述されています。

2.2.3 Microsoft* Visual Studio* 2005 の管理者権限に関するメッセージダイアログ

Microsoft* Visual Studio* 2005 を使用している場合、Microsoft* Windows Vista* またはそれ以降の Microsoft* Windows* にインストール中、次のようなダイアログが表示されることがあります。

このダイアログが表示された場合は、[常にこのメッセージを使用する] ボックスをオンのままにして、[続行] ボタンをクリックします。[Visual Studio の終了] を選択したり、何もしない場合 (このメッセージは 2 分後にタイムアウトします)、コンパイラー統合のインストールは完了しません。

詳細は、「Microsoft* Windows Vista* でのインストール」を参照してください。

2.3 製品の変更、更新、削除

Windows* のコントロールパネルの [プログラムの追加と削除] でインストールまたは削除する製品コンポーネントを変更します。

製品のアップデート・バージョンをインストールする際、古いバージョンを最初にアンインストールする必要はありません。複数のバージョンのコンパイラーをインストールし、その中から選択して使用することができます。新しいバージョンのコンパイラーを削除した場合、以前のバージョンの Microsoft* Visual Studio* への統合を再インストールする必要があります。

2.4 インストール先フォルダー

インストール・フォルダーの構成を以下に示します。一部含まれていないフォルダーもあります。

- C:\Program Files\Intel\ComposerXE-2011
  - bin
    - ia32
    - ia32_intel64
    - intel64
  - compiler
    - include
      - ia32
      - intel64
    - lib
      - ia32
      - intel64
  - debugger
  - Documentation
  - Help
  - ipp
    - bin
    - demo
    - include
    - interfaces
    - lib
    - tools
  - mkl
    - benchmarks
    - bin
    - examples
    - include
    - interfaces
    - lib
    - tests
    - tools
  - tbb

        - bin
        - examples
        - include
        - lib
    - redist
    - Samples
    - VS Integration

bin、include および lib 配下のフォルダーは次のとおりです。

- ia32: IA-32 上で動作するアプリケーションのビルドに使用するファイル
- intel64: インテル® 64 上で動作するアプリケーションのビルドに使用するファイル
- ia32_intel64: IA-32 上での実行用のコンパイラー。インテル® 64 上で動作するアプリケーションをビルドします。

英語以外の Windows* システムにインストールする場合、Program Files フォルダー名が異なる場合があります。インテル® 64 アーキテクチャー・システムでは、フォルダー名は Program Files (X86) またはそれに相当する名前です。

デフォルトでは、アップデートによって既存のディレクトリーの内容が置換されます。最初のアップデートをインストールするときに、以前のインストールとは別に新しいアップデートをインストールして、システムに両方のファイルを残すオプションを選択できます。両方を残すオプションを選択した場合、古いアップデートのトップレベルのフォルダー名は ComposerXE-2011.*nnn* (*nnn* はアップデート番号) に変更されます。

2.5 インストールの既知の問題

2.5.1 Microsoft* Visual Studio* 2010 用ドキュメントをインストールするための追加ステップ

Microsoft* Visual Studio* 2010 がインストールされているシステムにインテル® C++ Composer XE 2011 を初めてインストールするとき、Visual Studio* 2010 のドキュメントの「ローカルストア」を初期化するかどうか確認するメッセージが表示されます (初期化を行っていない場合)。「ヘルプ ライブラリ マネージャー」によってインテル® C++ Composer XE 2011 ヘルプ・ドキュメントが Visual Studio* 2010 内に登録されます。「ヘルプ ライブラリ マネージャー」のインストール・ウィザードの説明に従って、Visual Studio* 2010 用のインテル® C++ Composer XE 2011 ヘルプ・ドキュメントをインストールします。

このステップは 1 回のみ実行する必要があります。将来インテル® C++ Composer XE 2011 のアップデートをインストールするときに、「ヘルプ ライブラリ マネージャー」を使用してドキュメントを再登録する必要はありません。

詳細は、http://msdn.microsoft.com/ja-jp/library/dd264831.aspx を参照するか、「ヘルプ ライブラリ マネージャー」で検索してください。

2.5.2 複数の Visual Studio* のバージョンを使用している場合のドキュメントに関する問題

システムに Microsoft* Visual Studio* 2005 と 2008 の両方をインストールしていてインテル® C++ Composer XE 2011 を両方のバージョンに統合した場合、いずれかのバージョンを削除する

と両方のバージョンから統合されていたインテル® C++ Composer XE 2011 ドキュメントが削除されます。

ドキュメントを再インストールするには、以下の操作を行います。

1. コントロール パネルを使用して製品を選択します。
   - Windows* XP の場合: [コントロール パネル] - [プログラムの追加と削除] を選択します。
   - Windows* 7 の場合: [コントロール パネル] - [プログラムと機能] を選択します。
   - Windows Vista* の場合: [コントロール パネル] - [プログラム] を選択します。
2. 製品を選択した後、[変更と削除] ボタンをクリックします。
3. [コンポーネントの選択] ダイアログボックスで、[統合ドキュメント] の選択を解除します。ドキュメントが削除されます。
4. ステップ 1 と 2 を繰り返します。
5. [コンポーネントの選択] ダイアログボックスで、[統合ドキュメント] を選択します。ドキュメントが再インストールされます。

3 インテル® C++ コンパイラー

このセクションでは、インテル® C++ コンパイラーの変更点、新機能、および最新情報をまとめています。

3.1 互換性

バージョン 11 では、IA-32 システムのデフォルトでのコード生成において、アプリケーションを実行するシステムでインテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) がサポートされていると仮定するように変更されました。詳細は、下記を参照してください。

3.2 新機能と変更された機能

インテル® C++ コンパイラー XE 12.0 では、次の機能が新たに追加または大幅に拡張されています。これらの機能に関する詳細は、Visual Studio* に統合されたドキュメントを参照してください。

- インテル® Cilk™ Plus。インテル® C++ コンパイラー向けのこの言語拡張を使用することで、新規および既存のソフトウェアを簡単に並列化できます。
- ガイド付き自動並列化
- C++0x からの機能
  - 右辺値参照
  - 標準的なアトミック演算
  - “Windows* C++” モードでの C99 の 16 進浮動小数点定数のサポート
  - 右角括弧
  - 拡張 friend 宣言
  - 混在した文字列リテラルの結合
  - `long long` のサポート
  - 可変引数マクロ

    - スタティック・アサーション
    - auto 型変数
    - extern テンプレート
    - __func__事前定義済み指定子
    - 式の型宣言 (decltype)
    - ユニバーサル文字名
    - 強い型付けの列挙型
    - ラムダ
- より高速でやや精度が低い算術ライブラリー関数を使用するためのオプション
- プロセッサーのモデルや製造元に関係なく一貫した結果を返す算術ライブラリー関数を使用するためのオプション

3.2.1 スタティック・セキュリティー解析機能 (旧: ソースチェッカー) にはインテル® Inspector XE が必要

バージョン 11.1 の「ソースチェッカー」機能が拡張され、「スタティック・セキュリティー解析」に名称が変更されました。スタティック・セキュリティー解析を有効にするためのコンパイラー・オプションはバージョン 11.1 と同じですが (例: `/Qdiag-enable:sc`)、解析結果がコンパイラー診断結果ではなく、インテル® Inspector XE で表示可能なファイルに出力されるようになりました。

3.2.2 Microsoft* Visual Studio* 2010 のサポート

本リリースでは、Microsoft* Visual Studio* 2010 をサポートしています。Visual Studio* 2010 のインテル® C++ 統合は、Visual Studio* 2005 または 2008 の統合とは大きく異なります。詳細は次の記事を参照してください: http://software.intel.com/en-us/articles/what-are-the-differences-in-ide-integration-of-visual-studio-2010-and-visual-studio-2005-2008/

3.2.3 インテル® C++ プロジェクト・ファイルの互換性

インテル® C++ プロジェクト・ファイル (`.icproj`) の形式がバージョン 12.0 で変更されました。インテル® C++ の古いバージョンで作成されたプロジェクトを開くと、プロジェクトの変換が必要である旨のメッセージが表示されます。バージョン 12 のプロジェクトを古いバージョンのインテル® C++ 統合で使用することはできません (ただし、古いバージョンのコンパイラーは、[ツール] > [オプション] > [Intel C++ (インテル(R) C++)] > [Compilers (コンパイラー)] から使用できます)。

3.3 新規および変更されたコンパイラー・オプション

コンパイラー・オプションの詳細に関しては、ドキュメントのコンパイラー・オプションのセクションを参照してください。

- /Qansi-alias-check
- /Qcilk-serialize
- /Qdiag-sc-dir
- /Qfp-trap
- /Qfp-trap-all

- /Qguide
- /Qguide-data-trans
- /Qguide-file
- /Qguide-file-append
- /Qguide-opts
- /Qguide-par
- /Qguide-vec
- /Qimf-absolute-error
- /Qimf-accuracy-bits
- /Qimf-arch-consistency
- /Qimf-max-error
- /Qimf-precision
- /Qintel-extensions
- /Qopt-args-in-regs
- /Qopt-matmul
- /Qpatchable-addresses
- /Qprof-value-profiling
- /Qprofile-functions
- /Qprofile-loops
- /Qprofile-loops-report
- /Qpar-runtime-control[n]
- /Qpar-runtime-control-
- /Qregcall
- /Qsimd[-]
- /Qzero-initialized-in-bss

廃止予定のコンパイラー・オプションのリストは、ドキュメントのコンパイラー・オプションのセクションを参照してください。

3.3.1 廃止予定のオプション

次の方法で廃止予定のすべてのコンパイラー・オプションを確認できます。

1) [スタート] メニューからコマンドプロンプトを開きます: [スタート] > [すべてのプログラム] > [Intel Parallel Studio XE 2011] > [Command Prompt (コマンドプロンプト)] > [Parallel Studio XE with Intel Compiler XE 2011 v12.0 (インテル(R) コンパイラー XE 12.0)] > [IA-32 Visual Studio xxx mode (IA-32 Visual Studio xxx モード)] を選択します。
2) 次のコマンドを実行します。

```
>> icl /? deprecate
```

3.4 その他の変更

3.4.1 ビルド環境コマンドスクリプトの変更

ビルド環境を構築するコマンド・ウィンドウ・スクリプトが使用する Microsoft* Visual Studio* バージョンを任意で指定できるよう変更されました。ビルド環境ウィンドウを開くのに、定義

済みのスタート・メニュー・ショートカットを使用していない場合は、次のコマンドを使用して適切な環境を構築してください。

```
"<install-dir>\bin\compilervars.bat" arch [vs]
```

arch はビルドする対象アーキテクチャーを指定します。次のいずれかの値を指定できます。

- `ia32`
- `ia32_intel64`
- `intel64`

vs は任意で指定します。次のいずれかの値を指定できます。*vs* が指定されていない場合は、コマンドライン統合用にインストール時に指定された Visual Studio* のバージョンがデフォルトで使用されます。

- `vs2010`
- `vs2008`
- `vs2005`

また、インテル® Visual Fortran Composer XE 2011 もインストールされている場合、このコマンドによりインテル® Visual Fortran Composer XE 2011 を使用する環境も構築されます。

スクリプトファイル名 iclvars.bat および ifortvars.bat は、以前のリリースとの互換性のために保持されています。

3.4.2 OpenMP* レガシー・ライブラリーの削除

本リリースでは、OpenMP* のレガシー・ライブラリーが削除されました。"互換性がある" ライブラリーのみ提供されます。

3.4.3 OpenMP* ライブラリーのデフォルトがダイナミック・リンクに変更

バージョン 11.0 より、デフォルトで OpenMP* アプリケーションはダイナミック OpenMP* ライブラリーにリンクされます。OpenMP* ライブラリーのスタティック・リンクを指定するには、`/Qopenmp-link:static` を指定します。スタティック・ライブラリーは廃止され、将来のリリースでは削除される可能性があります。

3.4.4 バージョン管理システムでのインテル® C++ プロジェクトの使用

プロジェクトがバージョン管理システム (例: Microsoft* Visual SourceSafe* や Microsoft* Visual Studio* Team Foundation Server など) で管理されている場合、プロジェクトでインテル® C++ プロジェクト・システムを使用するには追加のステップが必要です。このトピックについての詳細な記事は、http://software.intel.com/en-us/articles/tips-on-using-the-intel-c-compiler-with-source-code-control-software/ (英語) を参照してください。

3.5 既知の問題

3.5.1 コンパイラーの既知の問題

3.5.1.1 日本語ファイル名に関するコマンドライン診断表示の問題

コンパイル診断で日本語が含まれているファイル名は、ネイティブのインテル® 64 対応アプリケーション用コンパイラーを使用して、Windows* コマンドでコンパイルした場合に正しく表示されません。Visual Studio* を使用する場合やインテル® 64 対応アプリケーション用クロスコンパイラーまたは IA-32 対応アプリケーション用コンパイラーを使用する場合は、この問題は発生しません。

3.5.2 Visual Studio* の既知の問題

3.5.2.1 Visual Studio* 2010 では `/fp:precise` がデフォルトでオン

Visual Studio* 2010 で作成または変換されたプロジェクトでは、デフォルトで `/fp:precise` コマンドライン・オプションがオンになります。このオプションは、パフォーマンスを低下させるいくつかの最適化を無効にして、浮動小数点演算の一貫性を向上させる「浮動小数点モデル」を設定します。インテルのデフォルトである `/fp:fast` に戻すには、プロジェクトのプロパティー・ページで [C/C++] > [Code Generation (コード生成)] > [Floating Point Model (浮動小数点モデル)] を Fast に変更します。

3.5.2.2 Microsoft* Visual Studio* 2005 のカスタムビルド規則

インテル® C++ コンパイラーの Microsoft* Visual Studio* 2005 への統合は、Visual C++* のカスタムビルド規則機能をサポートしています (一部の機能は制限されています)。カスタムビルド規則を使用すると、ビルド処理にカスタムツールの呼び出しを追加できます。カスタムビルド規則に関する詳細は、Visual C++* のドキュメントを参照してください。

カスタムビルド規則は、Visual C++* プロジェクトをインテル・プロジェクトに変換する前に作成する必要があります。インテル・プロジェクトに変換した後にカスタムビルド規則を変更する場合は、いったんプロジェクトを Visual C++* プロジェクト・システムに変換してからカスタムビルド規則を変更し、再度インテル・プロジェクトに変換して戻します。

カスタムビルド規則に対して定義されている文字列のプロパティーを変更する場合、利用可能なマクロのリストにインテル固有のマクロ (`$(icInstallDir); $(icIDEInstallDir); $(icProjectExt); $(icProjectFileName)` など) は含まれていません。ただし、カスタムビルド規則のプロパティー値でインテル固有のマクロを使用することはできます。指定されたマクロは正しく機能します。

インテル® C++ コンパイラーの Visual Studio* への統合では、現在 Visual C++* のツールのビルド順序機能 (Visual C++* の [Tool Build Order (ツールのビルド順序)] ダイアログボックスから利用可能) をサポートしていません。つまり、ビルドステップの実行順序を変更することはできません。

3.5.2.3 Visual Studio* 2010 の言語パック

インテル® C++ Composer XE 2011 をインストールした後に Visual Studio* 2010 の新しい言語パックをインストールすると、[プロジェクト プロパティ] ダイアログにインテル® C++ コンパ

イラー固有のオプションが表示されなくなることがあります。その場合には、以下の手順を試してみてください。

1) “<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\Intel C++ Compiler XE 12.0\1033" ディレクトリーが存在する場合は、すべてのファイルを “<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\Intel C++ Compiler XE 12.0\<locale-ID>” にコピーします。

2) "<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\v100\1033\" が存在する場合は、すべてのファイルを “<program files>\MSBuild\Microsoft.Cpp\v4.0\Platforms\[Win32|x64]\PlatformToolsets\v100\<locale-ID>” にコピーします。

* <locale-ID> は言語パックを表します。

別の方法として、インテル® C++ Composer XE 2011 をアンインストールして、再インストールすることもできます。

3.5.3 インテル® Cilk™ Plus の既知の問題

1) スチールが行われた後、対応する _Cilk_sync の前に SEH 例外がスローされると、Microsoft* C++ 構造化例外処理 (SEH) は失敗します。

2) if/else 条件文の if ブロック内にある cilk_spawn はコンパイルエラーになることがあります。
例えば、次のコードをコンパイルするとインテル® C++ コンパイラーはエラーを発行します。

```
if (expr)
    cilk_spawn a();
else
    b();

test.cpp
test.cpp(12): エラー: 文を指定してください。
   else
   ^
```

この問題は、次のように "cilk_spawn" を {} で囲むことで回避できます。

```
if (expr) {
    cilk_spawn a();
} else
    b();
```

3.5.4 ガイド付き自動並列化の既知の問題

プログラム全体のプロシージャー間の最適化 (/Qipo) が有効な場合、単一ファイル、関数名、ソースコードの指定範囲に対してガイド付き自動並列化 (GAP) 解析は行われません。この問題を回避するには、/Qipo を無効にします。Visual Studio* では、[プロジェクト] > [プロパティ ページ] > [C/C++] > [Optimization (最適化)] > [Interprocedural Optimization (プロシージャー間の最適化)] を「No (いいえ)」に設定します。

3.5.5 スタティック・セキュリティー解析の既知の問題

3.5.5.1 仮想関数を含む C++ クラスに対する正しくないメッセージ

スタティック・セキュリティー解析機能を使用するためには、インテル® Inspector XE も必要です。

プログラムで仮想関数を含む C++ クラスが使用されている場合に、スタティック・セキュリティー解析は正しくない診断を多数出力します。場合によっては、診断結果の数が多すぎて結果ファイルが使用できないこともあります。

このような C++ ソース構造を使用しているアプリケーションでは、次のコマンドライン・スイッチを追加することで不要なメッセージを表示しないようにできます: `/Qdiag-disable:12020,12040` (Windows*) または `-diag-disable 12020,12040` (Linux*)。**このスイッチは、スタティック・セキュリティー解析の結果が作成されるリンクステップで追加する必要があります。**コンパイルステップで追加しただけでは十分な効果が得られません。Microsoft* Visual Studio* では、このスイッチを [プロパティ ページ] > [Linker (リンカー)] > [Command Line (コマンドライン)] に追加します。

ビルド仕様ファイルを使用してスタティック・セキュリティー解析を行う場合は、`-disable-id 12020,12040` スイッチを inspxe-runsc の呼び出しに追加します。
例:

```
inspxe-runsc -spec-file mybuildspec.spec -disable-id 12020,12040
```

この問題を含む作成済みのスタティック・セキュリティー解析結果がある場合は、インテル® Parallel Inspector XE の GUI でそのファイルを開いて、次の手順に従って不要なメッセージを非表示にすることができます。

- 不要なメッセージは “`Arg count mismatch` (引数の数の不一致)” と “`Arg type mismatch` (引数の型の不一致)” です。それぞれの問題に対して、次の手順を実行します。
- 問題フィルターで不要な問題の種類をクリックします。これにより、それ以外の問題が非表示になります。
- 問題セットの表で任意の問題をクリックします。
- Ctrl+A キーを押すとすべての問題を選択できます。
- 右クリックしてポップアップ・メニューから *[Change State (ステートの変更)]* > *[Not a problem (問題なし)]* を選択し、不要なすべての問題のステートを設定します。
- 問題の種類のフィルターを [All (すべて)] に戻します。
- 他の不要な問題の種類に対して、上記の手順を行います。

- [Investigated/Not investigated (調査済み/未調査)] フィルターを *[Not investigated (未調査)]* に設定します。このフィルターは最後の方にあるため、フィルターペインを下にスクロールしないと見えないことがあります。[Not a problem (問題なし)] ステートは [Not investigated (未調査)] と見なされるため、これで不要なメッセージが非表示になります。

4 インテル® インテグレーテッド・パフォーマンス・プリミティブ

このセクションでは、インテル® インテグレーテッド・パフォーマンス・プリミティブ (インテル® IPP) のこのバージョンでの変更点、新機能、および最新情報をまとめています。インテル® IPP についての詳細は、次のリンクを参照してください。

- **新機能:** インテル® IPP 製品ページ (http://www.intel.com/software/products/ipp/ (英語) を参照してください。
- **ドキュメント、ヘルプ、サンプル:** インテル® IPP 製品ページ (http://www.intel.com/software/products/ipp/) のドキュメントのリンクを参照してください。

4.1 インテル® インテグレーテッド・パフォーマンス・プリミティブ 7.0

4.1.1 新機能と変更された機能

- 256 ビットの AVX SIMD 命令セット向けの最適化拡張 (開発コード名が “Sandy Bridge” というインテル® プロセッサーで利用可能) が組み込まれています。
- 暗号化ドメイン (別途ダウンロードが必要。以下を参照。) とデータ圧縮 (ipp_bzip2 向けの CRC32) においてより多くの AES-NI 最適化が適用されており、AES-NI 命令をサポートしているプロセッサーでパフォーマンスが大幅に向上します。
- Windows* 版の IPP ライブラリーでは Microsoft* Visual Studio* 2010 をサポートしているため、IPP のヘルプファイルとプロジェクト・ファイルを Visual Studio* 2010 の IDE で利用できます。Visual Studio* 2005 および 2008 の IDE もサポートされています。
- 16s、32s、32f データ型の JPEG-XR (HD Photo) の正変換/逆変換のサポート、および 32s データ型の可変長符号 (VLC) のエンコード/デコード関数のサポートが追加されました。
- JPEG-XR (HD Photo) コーデックが、ピクセルの深さが 8、16、32 ビット整数および 16、32 ビット浮動小数点数のグレースケール、RGB、RGBA イメージ向け IPP UIC サンプル・フレームワークに追加されました。
- DMIP サンプルに Visual Studio* 2008 で使用可能な Microsoft* DSL (ドメイン固有言語) のアドオンが追加されました。
- 新しく interfaces ディレクトリーが追加されました。このディレクトリーには、ハイレベルのアプリケーション・コードのソースバイナリーとビルド前のバイナリーが含まれます。いくつかのよく使用されるデータ圧縮ライブラリー (bzip2、zlib and gzip など) が IPP ライブラリーでも使用できるように変更され、*interfaces* ディレクトリーに格納されています。
- 本リリースの一部として、新しい ipp_lzopack (データ圧縮) ライブラリーが interfaces ディレクトリーにあります。
- OpenMP* マルチスレッド・ライブラリーの使用により、ipp_zlib ライブラリーの一部がマルチスレッドに対応しました。

- 新しいディレクトリー構造により、インテル® IPP ライブラリーとインテル® コンパイラー製品の統合が単純化されました。この変更に伴い、ビルドスクリプト、makefile、Visual Studio* プロジェクト・ファイルの更新が必要になることがあります。
- これまでの "em64t" ディレクトリーが "intel64"ディレクトリー に変更されました。この変更に伴い、ビルドスクリプト、makefile、Visual Studio* プロジェクト・ファイルの更新が必要になることがあります。
- 32 ビットと 64 ビットのアーキテクチャー間で一貫性を保持するために、ライブラリー・ファイル名が正規化されました (例えば、すべての 64 ビットのライブラリー・ファイル名から "em64t" が削除されました)。この変更に伴い、ビルドスクリプト、makefile、Visual Studio* プロジェクト・ファイルの更新が必要になることがあります。
- ドメイン固有の "emerged"および "merged" スタティック・ライブラリー・ファイルは、参照を容易にするために 1 つにまとめられ (例: ippsemerged.lib + ippsmerged_t.lib ⇒ ipps_t.lib)、シングル・スレッド・スタティック・ライブラリーのサフィックスは "_l" に変更されました (マルチスレッド・スタティック・ライブラリーのサフィックスはこれまでと同様に "_t" です)。この変更に伴い、ビルドスクリプト、makefile、Visual Studio* プロジェクト・ファイルの更新が必要になることがあります。
- 本リリースには音声認識関数 (ippSR ドメイン) は含まれていません。このドメインは IPP 6.1 製品で継続してサポートされます。
- 本リリースではインテル® Itanium® アーキテクチャー (IA-64) をサポートしていないため、IA-64 用の最新リリースはインテル® MKL 6.1 です。
- SPIRAL 生成関数 (ippGEN ドメイン) は、別途ダウンロードにて配布されるようになりました。詳細は、下記の手順を参照してください。

4.1.2 ディレクトリー構成の変更

新しいディレクトリー構造により、インテル® コンパイラー製品の統合が単純化されました。これに伴い、インテル® インテグレーテッド・パフォーマンス・プリミティブを使用する既存のアプリケーションでビルドを行うためには、変更が必要になることがあります。

- 以前のバージョンのインテル® Parallel Composer またはインテル® C++ コンパイラーで [Select Build Components (ビルド・コンポーネントの選択)] ダイアログを使用してインテル® パフォーマンス・ライブラリーの設定を行った場合は、Microsoft* Visual Studio* で C++ プロジェクトを右クリックして [Intel C++I Composer XE 2011 (インテル(R) C++ Composer XE 2011)] > [Select Build Components (ビルド・コンポーネントの選択)] を選択して、表示されたダイアログで [OK] をクリックしてパスを更新します。
- [ツール] > [オプション] > [Intel C++ (インテル(R) C++)] > [Compilers (コンパイラー)] で使用するコンパイラーのバージョンを変更する場合は、上記の [Build Components (ビルド・コンポーネントの選択)] のパスも更新する必要があります。

4.2 別途ダウンロード可能なインテル® IPP 暗号化ライブラリー

インテル® IPP 暗号化ライブラリーは別途ダウンロード可能です。ダウンロードとインストールの手順については、http://software.intel.com/en-us/articles/download-ipp-cryptography-libraries/ (英語) を参照してください。

4.3 別途ダウンロード可能な SPIRAL 生成関数

*SPIRAL 生成関数*は、インテル® IPP ライブラリーの信号処理ドメインのサブセットです。この関数はバージョン 6.0 で追加され、その後 IPP ライブラリー全体のダウンロードサイズを縮小するために、別途インストールされるアドオン・ライブラリーに変更されました。ippGEN ドメインに含まれる関数はすべてプリフィックスが ippg です。次にこれらの関数を省略形で示します。

- ippgenGetLibVersion
- ippgDCT4_[32f|64f]
- ippgDCT4Init_[32f|64f]
- ippgDCT4InitAlloc_[32f|64f]
- ippgDCT4Free_[32f|64f]
- ippgDCT4GetSize_[32f|64f]

- ippgDFTFwd_CToC_[32fc|64fc]
- ippgDFTFwd_CToC_*_[32fc|64fc]
- ippgDFTInv_CToC_[32fc|64fc]
- ippgDFTInv_CToC_*_[32fc|64fc]

- ippgHartley_[32f|64f]
- ippgHartley_*_[32f|64f]

SPIRAL 生成関数は、コンパイラーと同様にインテル® ソフトウェア開発製品レジストレーション・センターからダウンロードできます。

4.4 インテル® IPP コードサンプル

インテル® IPP コードサンプルは、以下の Web サイトから入手できます。
http://www.intel.com/software/products/ipp/ (英語)

サンプルには、オーディオ/ビデオコーデック、画像処理、メディア・プレーヤー・アプリケーション、C++/C#/Java* からの呼び出し関数のソースコードが含まれています。サンプルのビルド方法についての説明は、各サンプルのインストール・パッケージの readme ファイルをご覧ください。

5 インテル® マス・カーネル・ライブラリー

このセクションでは、インテル® マス・カーネル・ライブラリーの変更点、新機能、および最新情報をまとめています。

5.1 本バージョンでの変更

1) BLAS
   - 一度に 2 つの行列-ベクトル積を計算するための新しい関数: [D/S]GEM2VU、[Z/C]GEM2VC
   - 混合精度の一般的な行列-ベクトル積を計算するための新しい関数: [DZ/SC]GEMV

- 2 つのスケールされたベクトルの和を計算するための新しい関数: *AXPBY
- 主要関数においてインテル® AVX による最適化: SMP LINPACK、レベル 3 BLAS、DDOT、DAXPY

2) LAPACK
   - 行優先順に対応した LAPACK 用の C インターフェイス
   - 1 つの新しい計算ルーチン (*GEQRFP)、2 つの新しい補助ルーチン (*GEQR2P と *LARFGP)、LAPACK 3.2.1 のアップデートを含む Netlib LAPACK 3.2.2 との統合
   - 主要関数においてインテル® AVX による最適化: DGETRF、DPOTRF、DGEQRF
3) PARDISO
   - マルチコア環境で問題と解のステップのパフォーマンスが向上
   - スパースの右辺の解算出と部分解ベクトルを出力する部分解算出の追加
   - アウトオブコア (OOC) 因数分解のパフォーマンスが向上
   - ゼロベース (C スタイル) の配列インデックスのサポート
   - 対称行列のスパースデータ構造で行列の対角上のゼロが不要
   - 新しい ILP64 PARDISO インターフェイスにより、LP64 ライブラリーにリンクされている場合に LP64 と ILP64 の両バージョンを使用可能
   - OOC モードでディスクにファイルを格納するのに必要なメモリーを並べ替え直後に予測可能
4) スパース BLAS
   - 形式変換関数ですべてのデータ型に対応 (単精度/倍精度の実数/複素数データ)、および関数の戻り値として並べ替えあり/並べ替えなし配列を使用可能
5) FFT
   - 新しい MPI FFTW 3.3alpha1 ラッパーによる新しいクラスター機能
   - クラスター FFT のロードバランスの改善によりパフォーマンスが向上
   - すべての 1D/2D/3D FFT においてインテル® AVX による最適化
   - SSE4.2 命令セットをサポートするすべてのシステムにおいて、基数が混在する単精度/倍精度データの 2D/3D FFT のパフォーマンスが向上
   - 2D/3D FFT における 2 つの実数配列として表される分割複素数データのサポート
   - 長さが大きな素数である 1D 複素数-複素数変換のサポート
   - クラスター 1D 複素数変換のハイブリッド並列化 (MPI + OpenMP)、および (MPI プロセス数の倍数である) ベクトル長のパフォーマンスの向上
6) VML
   - (ax+b)/(cy+d) の計算を行うための新しい関数。a、b、c、d はスカラー、x、y は実数ベクトル: v[s/d]LinearFrac()
   - 主要関数においてインテル® AVX による最適化
   - デノーマル数をゼロに設定するための新しいモデル、複素ベクトルのオーバーフロー・サポート、各 VML 関数に対して精度を設定するための追加パラメーターを含む新しい関数
7) VSL
   - 新しいサマリー統計関数群。基礎統計、共分散/相関関係、プールされたグループ/部分/厳密な共分散/相関関係、分位数/変量分位数、外れ値検出アルゴリズム、欠測値をサポート
     - パフォーマンスが最適化されたアルゴリズム:欠測値をサポートするための MI アルゴリズム、厳密な共分散を計算するための TBS アルゴリズム、外れ値を検出するための BACON アルゴリズム、(変量データの) 分位数を計算するための ZW アルゴリズム、プールされた共分散を計算するための 1PASS アルゴリズム

- SFMT19937 基本乱数ジェネレーター (BRNG) のパフォーマンスが向上
- インテル® AVX による最適化: MT19937 と MT2203 BRNG

8) ドキュメント: Microsoft* Visual Studio* 2010 に統合される Microsoft* Help Viewer* 1.x 形式の製品ドキュメント
9) ランタイムにディスパッチされるダイナミック・ライブラリーの追加により、ランタイムに検出された CPU またはライブラリー関数呼び出しに応じて、依存性のあるライブラリーを動的にロードする単一のインターフェイス・ライブラリーへのリンクが可能
10) 新しいディレクトリー構造により、インテル® MKL ライブラリーとインテル® Parallel Studio XE 製品ファミリーの統合が単純化され、これまでの "em64t" ディレクトリーが "intel64" ディレクトリーに変更
11) 本リリースではインテル® Itanium® アーキテクチャー (IA-64) をサポートしていないため、IA-64 用の最新リリースはインテル® MKL 10.2
12) スパースソルバー機能をインテル® MKL のコア・ライブラリーに完全統合。また名前に "solver" を含むライブラリーを製品から削除

5.2 権利の帰属

エンド・ユーザー・ソフトウェア使用許諾契約書 (End User License Agreement) で言及されているように、製品のドキュメントおよび Web サイトの両方で完全なインテル製品名の表示 (例えば、"インテル® マス・カーネル・ライブラリー") とインテル® MKL ホームページ (www.intel.com/software/products/mkl (英語)) へのリンク/URL の提供を正確に行うことが最低限必要です。

インテル® MKL の一部の基となった BLAS の原版は http://www.netlib.org/blas/index.html (英語) から、LAPACK の原版は http://www.netlib.org/lapack/index.html (英語) から入手できます。LAPACK の開発は、E. Anderson、Z. Bai、C. Bischof、S. Blackford、J. Demmel、J. Dongarra、J. Du Croz、A. Greenbaum、S. Hammarling、A. McKenney、D. Sorensen らによって行われました。LAPACK 用 FORTRAN 90/95 インターフェイスは、http://www.netlib.org/lapack95/index.html (英語) にある LAPACK95 パッケージと類似しています。すべてのインターフェイスは、純粋なプロシージャー用に提供されています。

インテル® MKL クラスター・エディションの一部の基となった ScaLAPACK の原版は http://www.netlib.org/scalapack/index.html (英語) から入手できます。ScaLAPACK の開発 は、L. S. Blackford、J. Choi、A. Cleary、E. D'Azevedo、J. Demmel、I. Dhillon、J. Dongarra、S. Hammarling、G. Henry、A. Petitet、K. Stanley、D. Walker、R. C. Whaley らによって行われました。

インテル® MKL の PARDISO は、バーゼル大学 (University of Basel) から無償で提供されている PARDISO 3.2 (http://www.pardiso-project.org (英語)) と互換性があります。

本リリースのインテル® MKL の一部の FFT 関数は、カーネギーメロン大学からライセンスを受けて、SPIRAL ソフトウェア生成システム (http://www.spiral.net/ (英語)) によって生成されました。本リリースのインテル® MKL の一部の FFT 関数は、ヒューストン大学からライセンスを受けて、UHFFT ソフトウェア生成システムによって生成されました。SPIRAL の開発は、Markus Püschel、José Moura、Jeremy Johnson、David Padua、Manuela Veloso、Bryan Singer、Jianxin Xiong、Franz Franchetti、Aca Gacic、Yevgen Voronenko、Kang Chen、Robert W. Johnson、Nick Rizzolo らによって行われました。

6 インテル® スレッディング・ビルディング・ブロック

本バージョンのインテル® スレッディング・ビルディング・ブロックの変更に関する詳細は、TBB ドキュメント・ディレクトリーの CHANGES というファイルを参照してください。

6.1 既知の問題

インテル® スレッディング・ビルディング・ブロックの本リリースに関する次の注意事項に留意してください。

6.1.1 ライブラリーの問題

- インテル® スレッド・チェッカーまたはインテル® スレッド・プロファイラーを使用した際により正確な結果を得るには、インテル® TBB とともに使用する前にそれらの製品の最新のアップデート・リリースをダウンロードしてください。
- 同じプログラムで連続してインテル® TBB と OpenMP* コンストラクトをともに使用していて、OpenMP* コードにインテル® コンパイラーを使用している場合、KMP_BLOCKTIME に小さな値 (例えば、20 ミリ秒) を設定するとパフォーマンスが向上します。この設定は、kmp_set_blocktime() ライブラリー呼び出しを使用して OpenMP* コード内で行うこともできます。KMP_BLOCKTIME および kmp_set_blocktime() の詳細は、コンパイラーの OpenMP* に関するドキュメントを参照してください。
- 一般に、アプリケーションやサンプルの非デバッグ (”リリース”) ビルドは、インテル® TBB ライブラリーの非デバッグバージョンとリンクし、デバッグビルドはインテル® TBB ライブラリーのデバッグバージョンとリンクします。Windows* システムでは、/MD オプションを使用してコンパイルした場合はインテル® TBB ライブラリーのリリース・ライブラリー、/MDd オプションを使用してコンパイルした場合はデバッグ・ライブラリーを使ってビルドしてください。他の組み合わせでは、ランタイムエラーが発生します。デバッグ・ライブラリーとリリース・ライブラリーの詳細については、製品の ”Documentation” サブディレクトリーに含まれているチュートリアルを参照してください。

7 インテル® Parallel Debugger Extension

このセクションでは、インテル® Parallel Debugger Extension の変更点、新機能、および最新情報をまとめています。

7.1 新機能

- インテル® Cilk™ Plus サポート
  - インテル® Cilk™ Plus プログラムをデバッグ時に再コンパイルなしでシリアル実行することができます。
  - インテル® Cilk™ Plus のワーカースレッドは、Visual Studio* デバッガーで明確にマークされます。
  - Cilk Plus スレッド・スタック・ウィンドウにスチールポイントと現在の Cilk Plus ワーカーが表示されます。
- スレッドウィンドウ

- データ共有検出の向上
- OpenMP* 3.0 のサポート
- Windows* OS の同期関数のサポート
- データ共有検出の解析パフォーマンスの向上

7.2 既知の問題

- Microsoft* Visual Studio* 2005 を使用している場合、6 つのインテル固有の例外を手動で有効に設定する必要があります。[デバッグ] > [例外] を選択し、[Win32 Exceptions] ツリーを展開して、以下の項目を有効にします。

```
a1a01db0 Intel Parallel Debugger Extension Exception 0
a1a01db1 Intel Parallel Debugger Extension Exception 1
a1a01db2 Intel Parallel Debugger Extension Exception 2
a1a01db3 Intel Parallel Debugger Extension Exception 3
a1a01db4 Intel Parallel Debugger Extension Exception 4
a1a01db5 Intel Parallel Debugger Extension Exception 5
```

  これは、プロジェクトごとに 1 回設定します。
- デバッグセッション中にインテルのデバッグ例外を無効にすると、Visual Studio* (Visual Studio* 2008 SP1 まで) がハングアップすることがあります。
- インテル® Parallel Debugger Extension を使用するには、OpenMP* ライブラリーが動的にリンクされている必要があります (デフォルト)。インテル® Parallel Debugger Extension を使用する場合、OpenMP* ライブラリーのスタティック・リンクを指定する `/Qopenmp-link:static` を使用しないでください。
- 並列デバッグを行う前に並列デバッグ・インストルメンテーションを有効にしてください (/debug:parallel スイッチ)。[プロジェクト] > [プロパティ ページ] > [構成プロパティ] > [C/C++] > [Debugging (デバッグ)] > [Enable Parallel Debug Checks (並列デバッグチェックを有効にする)] で [Yes (/debug:parallel) (はい (/debug:parallel))] を選択します。この設定を行わない場合、デバッガーはデータ共有イベントや再入可能な呼び出しでの中断を検出できません。
- Microsoft* Visual Studio* 2010 の場合:最初に Microsoft* Visual C++* でプロジェクトをビルドするように指定し、後でインテル® C++ コンパイラーを使用するように変更した場合、並列デバッグ環境のプロパティー設定 [Auto (自動)] が認識されるように、いったんソリューションを閉じてから再度開く必要があります。
- Microsoft* Visual Studio* 2008 を使用し、64 ビット・アプリケーションのデバッグを行う場合、Visual Studio* 2008 Service Pack 1 がインストールされている必要があります。
  - サービスパックがインストールされていない場合、Visual Studio* 2005 および 2008 での 64 ビット・アプリケーションのデバッグは、低メモリー領域にリンクされる場合のみ行うことができます。低メモリー領域にリンクされない場合、デバッグ対象が終了するまでイベントは表示されません。終了後、すべてのイベントがイベントウィンドウに表示されます。64 ビット・アプリケーションを適切にデバッグするには、[プロジェクト] > [プロパティ] > [Linker (リンカー)] > [Advanced (詳細)] でベースアドレスを `0x10000` に設定します。
- [Data Sharing Events (データ共有イベント)] ウィンドウで関数のローカル変数やヒープ変数が「???」と表示されます。

- SSE レジスターウィンドウが 64 ビット・アプリケーションで動作しません。ウィンドウに「???」と表示されます。
- スタティック・ローカル変数のフィルターがコンテキスト・メニューから正しく設定されません。
- 逆アセンブルビューで再入可能な呼び出しの検出が停止します。スタティック関数の場合、正しく動作しません。デザインモードでは、`{,,myapp.exe}my_extern_function` など、適切なコンテキスト演算子の後で関数を使用してください。
- デバッガー拡張ウィンドウの配置が "docked" から "floating" に変更されるとウィンドウは空のままです。この問題を回避するには、"docked" のままにしておくか、または配置の変更後にデバッグセッションを再起動します。
- デバッガー拡張では、Visual Studio* からアプリケーションを開始する必要があります。既存のプロセスへアタッチしている場合は動作しません。
- ウィンドウが非表示、あるいは閉じられた後に再度開かれた場合は、デフォルト (16 進) 設定に戻ります。

7.3 ドキュメント

インテル® Parallel Debugger Extension のドキュメントは、Microsoft* Visual Studio* の [ヘルプ] メニューから、または [Parallel Debugger Extension] ウィンドウがアクティブな状態で F1 キーを押して表示することができます。`debugger-documentation.htm` にある "HTML バージョン" へのリンクをクリックして表示することもできます。

8 著作権と商標について

最新の仕様をご希望の場合や製品をご注文の場合は、お近くのインテルの営業所または販売代理店にお問い合わせください。

本書で紹介されている注文番号付きのドキュメントや、インテルのその他の資料を入手するには、1-800-548-4725 (アメリカ合衆国) までご連絡いただくか、インテルの Web サイトを参照してください。

インテル・プロセッサー・ナンバーはパフォーマンスの指標ではありません。プロセッサー・ナンバーは同一プロセッサー・ファミリー内の製品の機能を区別します。異なるプロセッサー・ファミリー間の機能の区別には用いません。詳細については、http://www.intel.co.jp/jp/products/processor_number/ を参照してください。