インテル® C++ Composer XE 2013 SP1 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: 321414-005JA
2013 年 10 月 10 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 は統合的なソフトウェア開発ツールであり、各コンポーネントは異なるライセンスの下で提供されます。詳細は、パッケージに含まれるライセンスと本リリースノートの「著作権と商標について」を参照してください。

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。各コンポーネントの新機能の詳細は、各コンポーネントのリリースノートを参照してください。

1.1.1 Update 1

- Windows 8.1 のサポート
- インテル® C++ コンパイラー XE 14.0.1
- バージョン 14.0 日本語版の初期リリース
- インテル® マス・カーネル・ライブラリー 11.1 Update 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 8.0 Update 1
- インテル® スレッディング・ビルディング・ブロック 4.2 Update 1
- 新しい数値文字列変換ライブラリー:`libistrconv`
- インテル® MIC アーキテクチャー向けの新しいコンパイラー・オプション `/Qopt-gather-scatter-unroll (opt-gather-scatter-unroll)`
- 新しいコンパイラー・オプション `/Q[a]xMIC-AVX512(-[a]xMIC-AVX512)`
- インテル® Memory Protection Extensions (インテル® MPX) 向けの新しいコンパイラー・オプション `/Qcheck-pointers-mpx (-check-pointers-mpx)`
- インテル® Cilk™ Plus SIMD 対応関数 (例: `__declspec(vector(uniform(this)))` が “`uniform(this)`” に対応
- 新しい組込み関数 `_allow_cpu_features`

1.1.2 インテル® C++ Composer XE 2013 からの変更点

- オンライン・インストーラー
- インテル® C++ コンパイラー XE 14.0.0
- インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャーのサポート
- C++11 の機能 (-std=c++11)
- OpenMP* 4.0 の一部サポート
- インテル® Cilk™ Plus の変更
- __INTEL_COMPILER_UPDATE 事前定義済みマクロ
- ポインター型のアライメント修飾子
- フォルス・シェアリングを回避する変数定義属性
- 共有ライブラリーに含まれるコードをオフロードする際に -offload=mandatory オプションまたは -offload=optional オプションを指定してメインプログラムのリンクが必要
- _Cilk_shared の制限
- ほかの OpenMP* 機能とは関係なく特定の OpenMP* 4.0 機能を有効/無効にする /Qopenmp-offload および /Qopenmp-simd オプションの追加
- Silvermont† マイクロアーキテクチャー向けの /QxATOM_SSE4.2 オプションの追加
- インテル® MIC アーキテクチャー向けインテル® Debugger Extension 1.0
- インテル® マス・カーネル・ライブラリーがバージョン 11.1 にアップデート

- インテル® インテグレーテッド・パフォーマンス・プリミティブがバージョン 8.0 Update 1 にアップデート
- インテル® スレッディング・ビルディング・ブロックがバージョン 4.2 にアップデート
- インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャー向けインテル® Debugger Extension

1.2 製品の内容

インテル® C++ Composer XE 2013 SP1 Update 1 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 14.0.1。Linux* オペレーティング・システムを実行する IA-32、インテル® 64 アーキテクチャー・システム、またはインテル® Xeon Phi™ コプロセッサーで動作するアプリケーションをビルドします。
- インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャー向けインテル® Debugger Extension
- 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 8.0 Update 1
- インテル® マス・カーネル・ライブラリー11.1
- インテル® スレッディング・ビルディング・ブロック 4.2
- 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://intel.ly/q9JVjE (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)
- インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャー向けの開発/テスト:
    - インテル® Xeon Phi™ プロセッサー
    - インテル® メニーコア・プラットフォーム・ソフトウェア・スタック (インテル® MPSS)
    - オフロードコードのデバッグには Microsoft* Visual Studio* 2012 が必要です。
- Microsoft* Windows*、Microsoft* Windows* 7、Microsoft* Windows* 8、Microsoft* Windows* 8.1、Microsoft* Windows Server* 2008、Microsoft* Windows* HPC Server 2008、Microsoft* Windows Server* 2012 (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* 2010 または Microsoft* Visual Studio* 2008 SP1 が必要です。
    - Microsoft* Windows* 8、Microsoft* Windows* 8.1 および Microsoft* Windows Server* 2012 では、製品は「デスクトップ」環境にインストールされます。「Windows* 8 UI」アプリケーションの開発はサポートされていません。[4]

- 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* 2012 Standard Edition 以上 (C++ コンポーネントがインストールされていること)
    - Microsoft* Visual Studio* 2010 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* 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]
- 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++* Express 2012 for Windows Desktop
    - Microsoft* Visual C++* 2010 Express Edition
    - Microsoft* Visual C++* 2008 Express Edition
- インテル® 64 対応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++* Express 2012 for Windows Desktop
    - Microsoft* Windows* Software Development Kit for Windows* 8
- ドキュメントの参照用に Adobe* Reader* 7.0 以降

注

1. Microsoft* Visual Studio* 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。
4. インテル® C++ Composer XE は、Windows 8* UI アプリの開発をサポートしていません。インテルでは、ユーザーの皆様のご意見を常に参考にしています。例えば、Windows* 8 UI アプリケーションにインテル® C++ Compiler XE またはその他のインテル® ソフトウェア開発製品の機能を利用したい方は、インテル® プレミアサポート (https://premier.intel.com/) からご意見をお送りください。
   Windows* 8 UI アプリケーションの開発で、このサポートされていないインテル® ソフトウェア開発製品の機能をテストすることにご興味のある方は、「Experimenting with Intel C++ Composer XE for Windows and Windows 8 Store Apps」 (http://intel.ly/wLeXRo) (英語) をお読みください。

1.3.1 Visual Studio* 2008 のサポート終了予定

将来のリリースでは、Visual Studio* 2008 はサポートされなくなる予定です。

1.3.2 Windows XP* のサポート終了予定

将来のリリースでは、Windows* XP はサポートされなくなる予定です。

1.3.3 IA-64 アーキテクチャー (インテル® Itanium®) 開発のサポートを終了

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。

1.3.4 Windows Server* 2003 および Windows Vista* のサポートを終了

Windows Server* 2003 および Windows Vista* のサポートを終了しました。これらのオペレーティング・システムを使用している場合は、新しいバージョンへの移行を推奨します。

1.3.5 Visual Studio* 2005 のサポートを終了

Visual Studio* 2005 のサポートを終了しました。Visual Studio* 2005 を使用している場合は、新しいバージョンへの移行を推奨します。

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 (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® マイクロアーキテクチャーに非固有の特定の最適化は、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。この注意事項の適用対象である特定の命令セットの詳細は、該当する製品のユーザー・リファレンス・ガイドを参照してください。 改訂 #20110804

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

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。

日本語版は、インテル® C++ Composer XE 2013 SP1 初期リリースの後の Update で提供されます。

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://intel.ly/oZjpZs (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

インストール時に製品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 (http://registrationcenter.intel.com) 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 新しいオンライン・インストーラー (インテル® Composer XE 2013 SP1)

インテル® Composer XE 2013 SP1 では、デフォルトのダウンロード版インストール・パッケージが、サイズの小さいオンライン・インストーラーになりました。オンライン・インストーラーは、選択したパッケージを動的にダウンロードし、インストールします。このインストール・パッケージを利用するには、インターネット接続が必要です。また、インターネット・プロキシーを使用している場合は、プロキシーの設定が必要になることがあります。インターネットに接続されていないマシンにインストールする場合は、オンライン・パッケージではなくフル・パッケージを利用してください。

2.2 インテル® メニーコア・プラットフォーム・ソフトウェア・スタック (インテル® MPSS) のインストール

インテル® メニーコア・プラットフォーム・ソフトウェア・スタック (インテル® MPSS) は、インテル® Composer XE 2013 SP1 Windows* 版 (インテル® MIC アーキテクチャー対応) のインストール前またはインストール後にインストールできます。

最新バージョンのインテル® MPSS を使用することを推奨します。

ユーザー空間およびカーネルドライバーのインストールに必要な手順については、インテル® MPSS のドキュメントを参照してください。

2.3 インテル® Software Manager

インテル® Software Manager は、製品アップデートの配信方法を簡素化し、現在インストールされているすべてのインテル® ソフトウェア製品のライセンス情報とステータスを表示します。

将来の製品設計の参考のため、製品使用状況に関する匿名情報をインテルに提供する、インテル® ソフトウェア向上プログラムに参加できます。このプログラムは、デフォルトで無効になっていますが、インストール中または後から有効にして参加できます。参加はいつでも取りやめることができます。詳細は、http://intel.ly/SoftwareImprovementProgram (英語) を参照してください。

2.4 インストール前の準備

2.4.1 64 ビット・アプリケーション用の Visual Studio* の設定

Microsoft* Visual Studio* 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* 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 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。

2.5 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。

2.5.1 PATH 環境変数の変更によるコマンドシェル (cmd.exe) への一時的な影響

Windows* 7 または Windows* 8 では、インストーラーが PATH 環境変数に項目を追加すると、PATH の長さが非常に長くなり (2000-4000 文字)、システムを再起動するまで Windows* コマンドプロンプト (cmd.exe) が動作しなくなることがあります。システムを再起動しても同じ問題が発生する場合は、テクニカルサポート までお問い合わせください。

2.5.2 サイレント・インストール

自動インストール、「サイレント」インストール機能についての詳細は、
http://intel.ly/nKrzhv (英語) を参照してください。

2.5.3 クラスターでのインストール

インストールするマシンに Microsoft* Compute Cluster Pack のライセンスがあり、クラスターメンバーの場合、「フル・インストール」を選択すると、そのクラスターのアクセス可

能なすべてのノードに製品がインストールされます。「カスタム・インストール」を選択すると、現在のノードのみにインストールするオプションを選択できます。

2.5.4 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://intel.ly/pjGfwC (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。

2.6 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。

2.7 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\Composer XE 2013 SP1
  - bin
    - ia32
    - ia32_gfx
    - ia32_intel64
    - intel64
    - intel64_mic
    - sourcechecker
  - compiler
    - include
      - cilk
      - ia32
      - intel64
      - mic
    - lib
      - ia32
      - intel64
      - mic
        - locale
          - en_US
          - ja_JP
    - perf_headers
      - C++
  - Debugger
    - gdb
      - LICENSES
      - src

- target
  - mic
    - bin
    - lib
    - share
      - man
        - man1
  - w64_mic
    - bin
    - include
      - gdb
    - lib
    - share
      - gdb
        - python
          - gdb
            - command
            - function
        - syscalls
      - info
      - locale
      - man
        - man1
- debuggerextension
  - mic
    - scripts

- Documentation
  - en_US
    - compiler_c
      - cl
    - debugger
      - gdb
        - pdf
    - gs_resources
    - ipp
      - get_started_files
      - ipp_userguide
      - tutorials
    - mkl
      - get_started_files
      - mkl_userguide
      - tutorials
    - ssadiag_docs
    - tbb
      - get_started_files
      - html
      - tutorial
    - tutorials
      - cmp_gap_c
      - cmp_thd_c
      - cmp_vec_c
  - msvhelp
    - 1033

                - compiler_c
                - ipp
                - mkl
                - ssadiag
                - tbb
        - vshelp
            - intel.cppprodocs
            - intel.cprocompilerdocs
            - intel.ippdocs
            - intel.mkldocs
            - intel.sssadiag
            - intel.tbbdocs
- Help
- ipp
    - bin
    - examples
    - include
    - interfaces
    - lib
    - tools
- mkl
    - benchmarks
    - bin
    - examples
    - include
    - interfaces
    - lib
    - tests
    - tools
- redist
    - ia32
        - compiler
            - 1033
            - irml
            - irml_c
        - ipp
            - 1033
        - mkl
            - 1033
        - tbb
            - vc_mt
            - vc9
            - vc10
            - vc11
    - intel64
        - compiler
            - 1033
            - irml
            - irml_c
        - ipp
            - 1033
        - mkl
            - 1033

- tbb
  - vc_mt
  - vc9
  - vc10
  - vc11

- Samples
  - en_US
    - C++
    - ipp
    - mkl
- tbb
  - bin
  - examples
  - include
    - serial
      - tbb
    - tbb
      - compat
      - internal
      - machine
  - lib
    - ia32
      - vc_mt
      - vc9
      - vc10
      - vc11
    - intel64
      - vc_mt
      - vc9
      - vc10
      - vc11
- VS Integration
  - C++
    - VS2008

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 Composer XE 2013.*nnn* (*nnn* はアップデート番号) に変更されます。

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++ Composer XE 2013 SP1 には、インテル® C++ コンパイラー XE 14.0 が含まれています。このバージョンでは、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、ドキュメントを参照してください。

- インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャーのサポート
- C++11 の機能 (-std=c++11)
  - 初期化子リストの完全実装(N2672 と N3217 を参照)
  - インライン名前空間の完全実装(N2535 を参照)
  - 非スタティック・データ・メンバー初期化子の完全実装(N2756 を参照)
  - 一般化された定数式の完全実装(N2235 を参照)
  - 無制限共用体の完全実装(N2544 を参照)
  - コンストラクターのデリゲート(N1986 を参照)
  - *this の rvalue 参照(N2439 を参照)
  - raw 文字列(N2442 を参照)
  - ラムダから関数ポインターへの変換
  - 暗黙の移動コンストラクターと代入演算子(N3053 を参照)
  - __bases および __direct_bases 型特性
  - クラス定義で状況依存キーワード "final" を、メンバー関数宣言で "final"と "override" を使用可能(N2928、N3206、N3272 を参照)
  - "noexcept" 指定子と演算子の完全実装(N3050 を参照)。noexcept の遅延インスタンス化を含む (Core issue 1330)。
- OpenMP* 4.0 の一部サポート
- インテル® Cilk™ Plus の変更
- __INTEL_COMPILER_UPDATE 事前定義済みマクロ
- ポインター型のアライメント修飾子
- フォルス・シェアリングを回避する変数定義属性

3.2.1 新しい数値文字列変換ライブラリー `libistrconv` (Update 1)

新しい数値文字列変換ライブラリー `libistrconv` は、ASCII 文字列と C データ型の変換を行う、パフォーマンスが最適化されたルーチン群を提供します。新しい API は、`"istrconv.h"` ヘッダーファイルで宣言されています。

詳細は、コンパイラー・ドキュメントを参照してください。

3.2.2 新しい組込み関数 `_allow_cpu_features` (Update 1)

新しい組込み関数 `_allow_cpu_features([xxx][,xxx])` が `immintrin.h` に追加されています。この組込み関数は、特定の最適化が行えるように、この組込み関数に続くコード領域を特定の機能を備えたプロセッサー向けにできることをコンパイラーに知らせます。

注: この組込み関数のサポートはまだ準備段階です。指定したコード領域ですべてのコンパイラーの最適化フェーズが実行されるわけではありません。

詳細は、コンパイラー・ドキュメント、コードサンプル、「New intrinsic _allow_cpu_features support」(英語) の記事を参照してください。

3.2.3 インテル® Cilk™ Plus SIMD 対応関数 (例: `__declspec(vector(uniform(this)))` が "`uniform(this)`" に対応 (Update 1)

これは、C++ クラスで SIMD 対応関数を使用するための機能です。同一のクラス・オブジェクトが SIMD 対応メンバー関数を呼び出す場合、呼び出し先の SIMD 宣言で明示的に "`uniform(this)`" 節を指定することでパフォーマンスが向上する可能性があります (呼び出し先で "`this`" キーワードが使用されているかどうかに依存します)。使用モデルは、仮引数に適用される uniform 節と同じです。

詳細は、コンパイラー・ドキュメントを参照してください。

3.2.4 OpenMP* 4.0 機能のサポート (インテル® Composer XE 2013 SP1)

インテル® Composer XE 2013 SP1 は、OpenMP* 4.0 の一部の機能をサポートしています。OpenMP* 4.0 仕様 (http://openmp.org) で定義されている、次の機能のサポートが追加されています。

- TEAMS プラグマ、宣言子、節
- DISTRIBUTE プラグマ、宣言子、節
- SIMD プラグマ、宣言子、節
- コプロセッサー (またはデバイス) 用の TARGET プラグマ、宣言子、節
- `#pragma omp taskgroup` 構造
- アトミック節 `seq_cst`
- 6 つの新しい形式のアトミックなキャプチャーと更新:
  - `Atomic swap: {v = x; x = expr;}`
  - `Atomic update: x = expr binop x;`
  - `Atomic capture 1: v = x = x binop expr;`
  - `Atomic capture 2: v =x = expr binop x;`
  - `Atomic capture 3: {x = expr binop x; v = x;}`
  - `Atomic capture 4: {v = x; x = expr binop x;}`
- `proc_bind(<type>)` 節。<type> は "spread"、"close"、または "master"。
- OMP_PLACES 環境変数
- OMP_PROC_BIND 環境変数
- `omp_get_proc_bind()` API

詳細は、http://intel.ly/W7CHjb (英語) を参照してください。

3.2.5 インテル® C++ Composer XE 2013 SP1 のインテル® Cilk™ Plus の変更点

インテル® C++ Composer XE 2013 SP1 では、インテル® Cilk™ Plus の次の新機能が追加されています。

- gcc と OpenMP* のベクトル関数の実装との互換性を向上するため、SIMD 対応関数の実装が変更されています。この変更により、インテル® C++ コンパイラーの以前のバージョン (13.1 以前) とバイナリー互換性がなくなります。インテル® C++ コンパイラー 14.0 で SIMD 対応関数を使用するすべてのコードをリビルドするか、以前の実装を使用するには /Qvecabi:legacy コンパイラー・オプションを指定する必要があります。
- 新しい乗算レデューサーが `cilk/reducer_opmul.h` で定義されています。
- ビット単位のリダクション操作をサポートするため、次の 3 つの新しい配列表記のリダクション組込み関数が追加されています。

- __sec_reduce_and
- __sec_reduce_or
- __sec_reduce_xor

3.2.6 仮定されるデータ・アライメントを指定するポインターとポインター型の新しい属性

インテル® Composer XE 2013 SP1 では、__declspec(align_value(N)) と __attribute__((align_value(N))) が追加されています。これらの属性が指定されたポインター型では、コンパイラーは指定されたアライメント “N” を仮定できます。次に例を示します。

```
typedef float float_a16
__attribute__((align_value (16)));

void foo(float_a16 *restrict dest, float_a16 *restrict src){
```

上記のコードは、src 引数と dest 引数がユーザーによって 16 バイト境界でアライメントされるべきであることをコンパイラーに知らせます。

3.2.7 フォルス・シェアリングを回避する変数定義属性

インテル® Composer XE 2013 SP1 では、__declspec(avoid_false_share)/__attribute__((avoid_false_share)) と __declspec(avoid_false_share(identifier))/__attribute__((avoid_false_share(identifier))) が追加されています。ほかの変数とのフォルス・シェアリングを回避するため、これらの属性が指定された変数で、コンパイラーは適切なパディングを追加するか、アライメントします。identifier が指定されている場合、同じ identifier の変数を除くほかの変数とのフォルス・シェアリングを回避するため、identifier が指定された変数はパディングが追加されるか、アライメントされます。これらの属性は、関数スコープ、グローバルスコープ、名前空間スコープの変数定義で指定します。関数スコープで指定する場合、identifier の有効範囲は現在の関数になります。名前空間スコープやグローバルスコープで指定する場合、identifier の有効範囲は現在のコンパイル単位になります。

3.2.8 インテル® Composer XE 2013 SP1 の新しい __INTEL_COMPILER_UPDATE 事前定義マクロ

新しい __INTEL_COMPILER_UPDATE 事前定義マクロにより、使用しているインテル® コンパイラーのマイナー・アップデート番号を取得できるようになりました。例えば、バージョン 14.0.2 の場合、このマクロは “2” に前処理されます。

3.2.9 スタティック解析機能 (旧:「スタティック・セキュリティー解析」または「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、「スタティック解析」に名称が変更されました。スタティック解析を有効にするコンパイラー・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。

3.2.10 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 14.0 (インテル® Composer XE 2013 SP1) 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 14.0 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。

3.3.1 インテル® Memory Protection Extensions (インテル® MPX) 向けの新しいコンパイラー・オプション /Qcheck-pointers-mpx (-check-pointers-mpx) (Update 1)

このオプションを指定すると、コンパイラーは、インテル® Memory Protection Extensions (インテル® MPX) を使用してポイントチェッカーのパフォーマンスを高速化するコードを生成します。ターゲット・プラットフォームがインテル® MPX をサポートしていない場合、ポインターチェッカーは通常の速度で動作します。インテル® MPX の詳細は、「Introduction to Intel® Memory Protection Extensions」 (英語) を参照してください。

新しいオプションの詳細は、ドキュメントの「コンパイラー・オプション」を参照してください。

3.3.2 インテル® アドバンスト・ベクトル・エクステンション 512 (インテル® AVX-512) 命令用の新しいコンパイラー・オプション /Q[a]xMIC-AVX512(-[a]xMIC-AVX512) (Update 1)

このオプションを指定すると、インテル® プロセッサー向けのインテル® AVX-512 の基本命令、競合検出命令、指数および逆数命令、プリフェッチ命令、および CORE-AVX2 で有効になる命令を生成します。インテル® AVX-512 命令対応のインテル® プロセッサー向けに最適化します。

詳細は、ドキュメントの「コンパイラー・オプション」を参照してください。新しい命令の詳細は、「AVX-512 instructions」 (英語) を参照してください。

3.3.3 インテル® MIC アーキテクチャー向けの /Qopt-gather-scatter-unroll(-opt-gather-scatter-unroll) (Update 1)

このオプションを使用して、インテル® MIC アーキテクチャーの集約 (Gather) と分散 (Scatter) ループに対して別のループ・アンロール・シーケンスを指定できます。このオプションは、インテル® MIC アーキテクチャー向けインテル® 64 アーキテクチャーでのみ利用できます。

詳細は、ドキュメントの「コンパイラー・オプション」を参照してください。

3.3.4 インテル® Composer XE 2013 SP1 の新規および変更されたコンパイラー・オプション

- /Qopenmp-offload[-]
- /Qopenmp-simd[-]
- /QxATOM_SSE4.2
- /QxATOM_SSSE3
- /Qvecabi:<arg>
- /Qpar
- /pdbfile[:filename]
- /nopdbfile
- /Wpch-messages[-]
- /Qmic
- /Qoffload[:keyword]
- /Qoffload-attribute-target:target-name
- /Qoffload-option, target, tool, “option-list”
- /Qopt-assume-safe-padding

- /Qopt-prefetch-distance:n1[,n2]
- /Qopt-streaming-cache-evict[:n]
- /Qopt-threads-per-core:n

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。

3.3.5 /Qopenmp-offload[-] と /Qopenmp-simd[-] の追加 (インテル® Composer XE 2013 SP1)

この 2 つのオプションにより、OpenMP* 4.0 の TARGET 機能と SIMD 機能を、ほかの OpenMP* 機能 (/Qopenmp で有効になる) とは別に有効/無効にできます。/Qopenmp を指定すると、/Qopenmp-offload と /Qopenmp-simd も指定されるため、これらの機能を利用できます。

3.3.6 /Wpch-messages[-] の追加 (インテル® Composer XE 2013 SP1)

インテル® Composer XE 2013 SP1 では、プリコンパイル済みヘッダーに関連する診断を有効または無効にする機能が追加されました。

3.3.7 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 2013] > [Command Prompt (コマンドプロンプト)] > [Parallel Studio XE with Intel Compiler XE v14.0 [Update xx] (インテル(R) コンパイラー XE 14.0 [Update xx])] > [IA-32/Intel(R) 64 Visual Studio xxx mode (IA-32/インテル(R) 64 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。

- `vs2012`
- `vs2010`
- `vs2008`

また、インテル® Visual Fortran Composer XE 2013 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* スタティック・ライブラリーの削除

本リリースでは、OpenMP* ランタイム・ライブラリーが削除されました。これらのランタイム・ライブラリーは動的にリンクしてください。

3.4.3 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://intel.ly/plmnpo (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 Internet Explorer *10 でオンライン・ドキュメントが表示されない問題*

Internet Explorer* 10 によってドキュメントで使用されているスクリプトがブロックされることがあります。この場合、空白のページが表示されるか、「スクリプトや ActiveX コントロールを実行しないよう、Internet Explorer で制限されています。」というエラーメッセージが表示されます。ドキュメントを表示するには、[ブロックされているコンテンツを許可] をクリックしてください。エラーメッセージが表示されない場合は、Internet Explorer で [ツール] > [インターネット オプション] > [セキュリティ] > [レベルのカスタマイズ] を選択し、安全だとマークされていないコンテンツをダウンロードする前にダイアログを表示するように設定します。

3.5.1.2 ドキュメントに含まれていない 10 進浮動小数点のステータスをチェックする関数

10 進の浮動小数点演算中に発生する例外を検出するには、次の浮動小数点例外関数を使用します。

関数	説明
fe_dec_feclearexcept	サポートされている浮動小数点例外をクリアします。
fe_dec_fegetexceptflag	浮動小数点ステータスフラグの実装定義の表現を格納します。
fe_dec_feraiseexcept	サポートされている浮動小数点例外を発行します。
fe_dec_fesetexceptflag	浮動小数点ステータスフラグを設定します。
fe_dec_fetestexcept	指定されている浮動小数点例外フラグのサブセットのうち、現在設定されているものを特定します。

10 進の浮動小数点例外関数は、fenv.h ヘッダーファイルで定義されています。

バイナリー形式の浮動小数点例外関数については、ISO C99 で説明されています。

DFP を使用してソースをコンパイルするには、プリプロセッサー・マクロ __STDC_WANT_DEC_FP__ を使用します。

3.5.1.3 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

3.5.3 Windows Server* 2012 で Visual Studio* 2012 のドキュメントを表示できない場合

Windows Server* 2012 で Visual Studio* 2012 のヘルプまたはドキュメントを表示できない場合、Microsoft* Internet Explorer* のセキュリティー設定を変更すると表示されるようになります。[ツール] > [インターネット オプション] > [セキュリティ] を選択して、[インターネット] ゾーンで [MIME スニッフィングを有効にする] および [アクティブ スクリプト] を有効にします。

3.5.3.1 MSVCP90D.dll (またはその他の Microsoft* ランタイム DLL) が見つからない

サンプルプロジェクト (および Microsoft* Visual C++* プロジェクト) を実行するときに Microsoft* Visual Studio* のランタイム DLL. が見つからない場合、ランタイムエラーが発生します。これは、マニフェスト・ファイルや SXS アセンブリーが見つからないことが原因です。この問題を解決するには、使用しているバージョンの Microsoft* Visual Studio* の redist フォルダー (デフォルトの場所は c:\program files[(x86)]\Microsoft Visual Studio X.X\VC\redist) に移動します。amd64、x86、Debug_NonRedist サブフォルダーで、必要なランタイムが含まれているフォルダーを探します (デバッグ・ライブラリーを探す場合は、ファイル名の最後が D のファイルが含まれているフォルダーを探します)。必要なランタイムが含まれているフォルダーが見つかったら、そのフォルダーの (.manifest ファイルを含む) すべての内容を、実行する .exe ファイルのあるフォルダーにコピーします。

3.5.3.2 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。

3.5.3.3 Visual Studio* 2010 の言語パック

インテル® C++ Composer XE 2013 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 14.0\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 14.0\<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]\PlatformTool
sets\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 2013 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。

3.5.4 インテル® Cilk™ Plus の既知の問題

- 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。

3.5.5 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。

3.5.6 スタティック解析の既知の問題

3.5.6.1 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・オプション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*)。**このオプションは、スタティック解析の結果が作成されるリンク時に追加する必要があります。**コンパイル時に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、このオプションを [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解析を行う場合は、-disable-id 12020,12040 オプションを 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解析結果がある場合は、インテル® 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ほうにあるため、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® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャー向けアプリケーションの開発

このセクションでは、インテル® Composer XE 2013 Windows* 版 (インテル® MIC アーキテクチャー対応) の変更点、新機能、および最新情報をまとめています。

4.1 インテル® Composer XE 2013 SP1 Windows* 版のインテル® MIC アーキテクチャー対応について

インテル® Composer XE 2013 SP1 Windows* 版 (インテル® MIC アーキテクチャー対応) は、インテル® Xeon Phi™ コプロセッサーで実行するコードの事前定義済みセクションを有効にすることにより、インテル® C++ Composer XE 2013 とインテル® Fortran Composer XE 2013 の機能セットが拡張されます。

コプロセッサーが利用可能な場合、コードのこれらのセクションはコプロセッサーで実行されます。コプロセッサーが利用できない場合は、ホスト CPU で実行されます。

本リリースノートでは、オフロード操作のターゲットについて、*コプロセッサー*と *ターゲット*という 2 つの用語を使用しています。

現在、インテル® Composer XE 2013 SP1 の次のコンポーネントでインテル® MIC アーキテクチャーをサポートしてします。

- インテル® C++ コンパイラーおよびインテル® Fortran コンパイラー
- インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャー向けインテル® Debugger Extension
- インテル® マス・カーネル・ライブラリー (インテル® MKL)
- インテル® スレッディング・ビルディング・ブロック (インテル® TBB)
- Visual Studio* IDE 統合

4.2 はじめに

インテル® 64 アーキテクチャー用とインテル® MIC アーキテクチャー用のコードは同じコンパイラーを使用して生成します。[スタート] メニューから [すべてのプログラム] > [Intel Parallel Studio XE 2013 (インテル(R) Parallel Studio XE 2013)] > [Command Prompt (コマンドプロンプト)] > [Parallel Studio XE with Intel Compiler XE v14.0 (インテル(R) コンパイラー XE 14.0)] を選択し、インテル® 64 モードのコマンドプロンプトを使用するか、または Microsoft* Visual Studio* IDE で x64 構成を使用して、インテル® 64 アーキテクチャー用のコンパイラーを使用します。

4.3 製品のドキュメント

インテル® Composer XE 2013 SP1 のインテル® MIC アーキテクチャーに関連するドキュメントには現在も修正が加えられています。ドキュメントの最新の情報については、Web サイト http://intel.ly/MxPFYx (英語) を参照してください。

4.4 インテル® マス・カーネル・ライブラリー (インテル® MKL)

インテル® MIC アーキテクチャーのサポートについての詳細は、「インテル® MKL」セクションを参照してください。

4.5 インテル® C++ コンパイラー

4.5.1 _Cilk_shared の制限 (インテル® Composer XE 2013 SP1)

- 仮想基本クラスで _Cilk_shared 属性を指定することはできません。
- 複数の基本クラスから _Cilk_shared 属性が指定されたクラスを派生させることはできません (複数の継承は許可されません)。
- _Cilk_shared 属性が指定されたクラスで仮想デストラクターを定義することはできません。
- _Cilk_shared 属性が指定されたクラスを別の _Cilk_shared クラスの基本クラスとして使用する場合、そのサイズが 8 の倍数になるように (必要に応じて、仮のフィールドを追加して) プログラマーが調整する必要があります。
- _Cilk_offload は、共有ライブラリー (DLL) を使うプログラムでは使用できません

4.5.2 共有ライブラリーに含まれるコードをオフロードする際に /Qoffload:mandatory オプションまたは /Qoffload:optional オプションを指定してメインプログラムのリンクが必要

オフロードには初期化処理が必要ですが、これはメインプログラムでのみ行うことができます。つまり、共有ライブラリーに含まれるコードをオフロードする場合、初期化処理が行われるように、メインプログラムもリンクしなければなりません。メインコードやメインプログラムへスタティック・リンクされたコードにオフロード構造が含まれる場合、これは自動で行われます。そうでない場合、/Qoffload:mandatory コンパイラー・オプションまたは /Qoffload:optional コンパイラー・オプションを指定して、メインプログラムをリンクする必要があります。

4.5.3 リンク時に検出されない見つからないシンボル

オフロード・コンパイル・モデルでは、インテル® MIC アーキテクチャーを対象とするバイナリーはダイナミック・ライブラリー (.so) として生成されます。ダイナミック・ライブラリーは、参照されている変数やルーチンをロード時に解決できるため、リンク時にこれらをすべて解決する必要はありません。この動作により、ロード時に解決できない一部の見つからない変数やルーチンを見逃してしまうことがあります。リンク時にすべての見つからないシンボルを識別して解決するには、次のコマンドライン・オプションを使用して未解決の変数をリストします。

```
/Qoffload-option,mic,compiler,"-z defs"
```

4.5.4 コンパイル時の診断の *MIC* タグ

ターゲット (インテル® MIC アーキテクチャー) とホスト CPU のコンパイルを区別できるようにコンパイラーの診断インフラストラクチャーが変更され、出力メッセージに *MIC* タグが追加されるようになりました。このタグは、インテル® MIC アーキテクチャー用のオフロード拡張を使用してコンパイルしたときに、ターゲットのコンパイル診断にのみ追加されます。

下記の例で、サンプル・ソース・プログラムは、ホスト CPU とターゲット (インテル® MIC アーキテクチャー) のコンパイルの両方で同じ診断を行っています。ただし、プログラムによっては、2 つのコンパイルで異なる診断メッセージが出力されます。新しいタグが追加されたことで、CPU とターゲットのコンパイルを容易に区別できることが分かります。

```
$ icl -c sample.c
sample.c(1): 警告 #1079:*MIC* 関数 "main" の戻り型は "int" でなければなりません。
  void main()
       ^

sample.c(5): 警告 #120:*MIC* 戻り値の型が関数の型と一致しません。
      return 0;
             ^

sample.c(1): 警告 #1079: 関数 "main" の戻り型は "int" でなければなりません。
  void main()
       ^

sample.c(5): 警告 #120: 戻り値の型が関数の型と一致しません。
      return 0;
```

4.5.5 ランタイム型情報 (RTTI) は未サポート

仮想共有メモリー・プログラミングでは、ランタイム型情報 (RTTI) はサポートされていません。特に、dynamic_cast<> と typeid() の使用はサポートされていません。

4.5.6 直接 (ネイティブ) モードにおけるランタイム・ライブラリーのコプロセッサーへの転送

インテル® メニーコア・プラットフォーム・ソフトウェア・スタック (インテル® MPSS) に、/lib 以下のインテル® コンパイラーのランタイム・ライブラリー (例えば、OpenMP* ライブラリー libiomp5.so) は含まれません。

このため、直接モード (例えば、コプロセッサー・カード上) で OpenMP* アプリケーションを実行する場合は、アプリケーションを実行する前にインテル® MIC アーキテクチャー OpenMP* ライブラリー (<install_dir>\compiler\lib\mic\libiomp5.so) のコピーをカード (デバイス名の形式は micN; 最初のカードは mic0、2 番目のカードは mic1、…) に (scp 経由で) アップロードする必要があります。

このライブラリーが利用できない場合、次のようなランタイムエラーが発生します。

```
/libexec/ld-elf.so.1:"sample" で要求された共有オブジェクト "libiomp5.so" が見つかりません。
```

libimf.so のような別のコンパイラー・ランタイムでも同様です。必要なライブラリーは、アプリケーションおよびビルド構成により異なります。

4.5.7 オフロード領域からの exit() の呼び出し

オフロード領域から exit() を呼び出すと、“オフロードエラー: デバイス 0 のプロセスがコード 0 で予想外に終了しました” のような診断メッセージが出力され、アプリケーションが終了します。

4.6 インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャー向けインテル® Debugger Extension

このセクションでは、インテル® Debugger Extension の変更点、新機能、カスタマイズ、および既知の問題をまとめています。インテル® Debugger Extension は、インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャー向けのコードのみサポートします。

4.6.1 機能

- オフロード拡張を使用して、コプロセッサーのネイティブ・アプリケーションとホスト・アプリケーションの両方をサポート
- 同時に複数のコプロセッサー・カードをデバッグ (オフロード拡張を使用)

4.6.2 インテル® Debugger Extension の使用

インテル® Debugger Extension は Microsoft* Visual Studio* IDE のプラグインです。Microsoft* Visual Studio* IDE で定義されたプロジェクトのデバッグを可能にします。インテル® Xeon Phi™ コプロセッサー向けアプリケーションは、ロードして実行することも、アタッチすることもできます。

インテル® Xeon Phi™ コプロセッサーにオフロードされるアプリケーションをデバッグするには、Microsoft* Visual Studio* IDE を起動する前に環境変数を設定する必要があります。インテル® メニーコア・プラットフォーム・ソフトウェア・スタック (インテル® MPSS) のバージョンに応じて、次の変数を設定します。

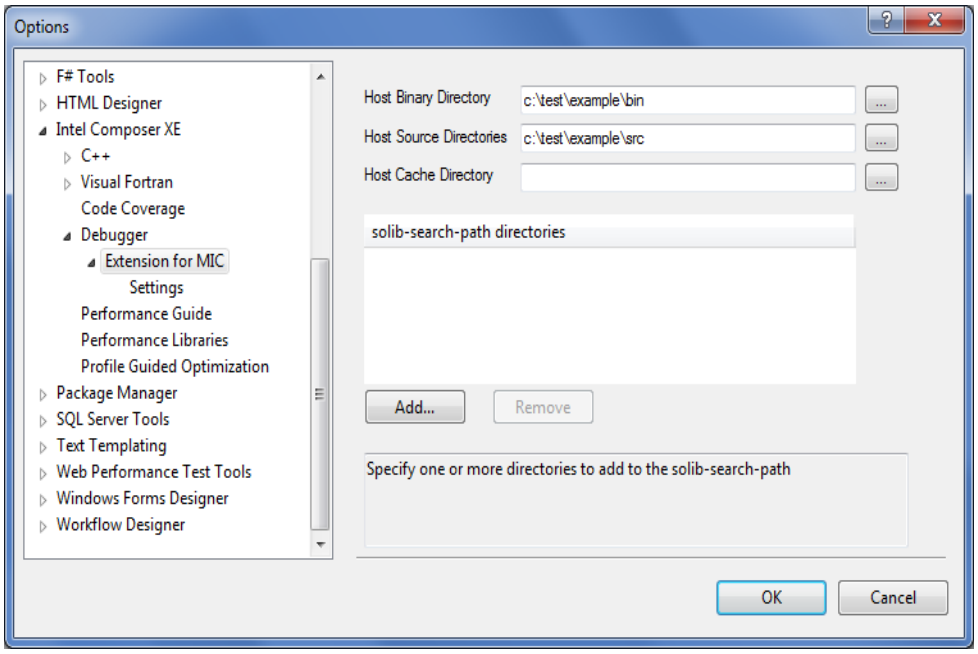
- インテル® MPSS 3.1:
  ```
  AMPLXE_COI_DEBUG_SUPPORT=TRUE
  MYO_WATCHDOG_MONITOR=-1
  ```
- インテル® MPSS 2.1:
  ```
  COI_SEP_DISABLE=FALSE
  MYO_WATCHDOG_MONITOR=-1
  ```

4.6.2.1 インテル® Xeon Phi™ コプロセッサー向けアプリケーションのロードと実行

これは、2 つのオフロードモデルのいずれかを使用しているアプリケーションでのみ行うことができます。インテル® Xeon Phi™ コプロセッサー向けプロジェクトを開き、[デバッグ] > [デバッグ開始] を選択してデバッグセッションを開始します。デバッグセッション中に表示されるコプロセッサーと情報 (スレッド、ブレークポイントなど) は、使用するオフロードモデルに依存します。つまり、オフロードモデルを使用していない場合、コプロセッサーでコードは実行されません。

4.6.2.2 インテル® Xeon Phi™ コプロセッサー上のプロセスにアタッチする

これは、インテル® Xeon Phi™ コプロセッサーでネイティブ実行されるアプリケーションでのみ行うことができます。デバッグするプロセスは、すでにコプロセッサーで実行されていなければなりません。
Microsoft* Visual Studio* IDE でプロセスバイナリーを指定します。[ツール] > [オプション...] を選択、次のように展開します。



[Host Binary Directory (ホストのバイナリー・ディレクトリー)] で、コプロセッサーでデバッグするバイナリー (のコピー) があるディレクトリーを指定します。ソースの検索に使用するルート・ディレクトリーは、[Host Source Directory (ホストのソース・ディレクトリー)] で指定できます。オプションで、コプロセッサー・アプリケーションの共有ライブラリーの検索パスを [solib-search-path directories (共有ライブラリーの検索パスのディレクトリー)] に追加できます。セマンティクスは GNU* GDB と同じです。

4.6.3 ドキュメント

インテル® デバッガーのドキュメントは、以下の場所にあります。

```
<install-dir>\Documentation\[en_US|ja_JP]\debugger\gdb\ ↵
              pdf\vsmigdb_config_guide.pdf
```

4.6.4 既知の問題と制限

- オフロードデバッグは Microsoft* Visual Studio* 2012 でのみサポートされています。
- 同じ Visual Studio* セッションで、インテル® MIC アーキテクチャーのプロセスに二度アタッチすると、[スレッド] ウィンドウにメインスレッド (ID=1) が表示されません。
  回避方法:
  メインスレッドに関する重要な情報は、[プロセス] ウィンドウで確認できます。
- [関数のブレークポイント] は動作しません。正常に動作しているように見えますが、実行を中断しません。
  回避方法:
  ソースファイルでブレークポイントを設定することができます。
- データ・ブレークポイントはサポートされていません。
- ブレークポイントのヒットカウント、条件、ヒット時の動作はサポートされていません。
- [逆アセンブル] ウィンドウでは、開始アドレスから 1024 バイトを超える範囲にスクロールすることはできません。
- [逆アセンブルアドレス] ウィンドウで式とアドレスはサポートされていません。
- インテル® MIC アーキテクチャー・アプリケーションの例外処理はサポートされていません。
- ブレークポイントの無効化は正しく動作しません。
  回避方法:
  ブレークポイントを削除して、再度設定します。

- アプリケーション実行中のブレークポイントの変更は正しく動作しません。変更されたかのように見えますが、変更が適用されません。
- インテル® MIC アーキテクチャーのネイティブ・アプリケーションの開始はサポートされていません。現在実行中のアプリケーションにアタッチすることはできます。
- 場合によっては、コプロセッサーで実行中のすべてのスレッドが [スレッド] ウィンドウに表示されないことがあります。
- Microsoft* Visual Studio* の [スレッド] ウィンドウには、スレッドの凍結、凍結解除、名前変更を行うコンテキスト・メニューがあります。これらのコンテキスト・メニューは、コプロセッサー上のスレッドでは正しく動作しません。
- オフロードコード領域のコード行でブレークポイントを設定すると、ホスト側とコプロセッサー側で有効になります。Microsoft* Visual Studio* の [ブレークポイント] ウィンドウでブレークポイントを展開すると、2 つのブレークポイントが表示されます。デバッグセッションを再起動すると、コプロセッサー側のブレークポイントは無効なブレークポイントとして残りますが、これは無視してください。
- 場合によっては、オフロードコードのデバッグ中、オフロード宣言子のソース位置で実行が停止することがありますが、実際の実行位置はライブラリー・ルーチン内です。その位置から実行を続行できます。
- インテル® 64 対応アプリケーションのみサポートされています。

5 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 8.0 の最新情報は、`<install dir>\Documentation\<locale>\ipp\ReleaseNotes.htm` にある製品のリリースノート (英語) を参照してください。

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://intel.ly/OG5IF7 (英語)) およびインテル® IPP リリースノート (http://intel.ly/OmWI4d (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://intel.ly/OG5IF7) のドキュメントのリンクを参照してください。

5.1 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://intel.ly/ndrGnR (英語) を参照してください。

5.2 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://intel.ly/pnsHxc (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。

6 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 (インテル® MKL) の変更点、新機能、および最新情報をまとめています。問題の修正については、http://intel.ly/OeHQqf (英語) を参照してください。

6.1 注意事項

注意事項についての詳細は、削除された機能に関するナレッジベースの記事 (英語) を参照してください。

- インテル® MKL では、インストールするコンポーネントを選択できるようになりました。PGI* コンパイラー、Compaq* Visual Fortran コンパイラー、SP2DP インターフェイス、BLAS95 および LAPACK95 インターフェイス、クラスターサポート (ScaLAPACK および Cluster DFT)、インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャーのサポートに必要なコンポーネントは、インストール時に明示的に選択しない限りインストールされません。
- インテル® MKL クラスター・コンポーネント (ScaLAPACK および Cluster DFT) では、アライメントされていない条件付き数値再現性 (CNR) は利用できません。
- Boost* uBLAS および Java* でのインテル® MKL の使用例は、製品パッケージからは削除され、以下の記事 (英語) からダウンロードすることができます。
  - How to use Intel MKL with Java*
  - How to use Boost* uBLAS with Intel MKL

6.2 本バージョンでの変更

6.2.1 インテル® MKL 11.1 Update 1 の新機能

- インテル® AVX-512 命令セットのサポート (特定の最適化のみ)
- BLAS:
  - インテル® アドバンスト・ベクトル・エクステンション (インテル® AVX) およびインテル® アドバンスト・ベクトル・エクステンション 2 (インテル® AVX2) 対応のすべての 64 ビットのインテル® プロセッサーにおいて、DSDOT のパフォーマンスが向上し、マルチスレッドをサポート
  - *TRSM で対角行列のデノーマル数の処理が向上
  - インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャーにおいて、小さな N と大きな M および K で SGEMM のパフォーマンスが向上
  - インテル® SSE4.2 以降対応のすべてのインテル® プロセッサーにおいて *HEMM の並列パフォーマンスが向上
  - インテル® SSSE3 以降対応のすべてのインテル® プロセッサーにおいて 64 ビットの *SYRK/*HERK の並列パフォーマンスが向上
  - インテル® SSE4.2 以降対応のすべてのインテル® プロセッサーにおいて 64 ビットの {D,S}SYRK のシリアル・パフォーマンスが向上
  - インテル® MIC アーキテクチャーにおいて DTRSM のパフォーマンスが向上
  - インテル® AVX 対応インテル® プロセッサー向けインテル® Optimized HPL Benchmark の runmultiscript 機能を拡張
  - インテル® MIC アーキテクチャーにおいてインテル® Optimized HPL Benchmark のパフォーマンスが向上
- LAPACK
  - 並列 LAPACK 関数 (OR/UN)M(QR/RQ/QL/LQ) のメモリー使用率が減少
  - LAPACK 関数のスタックメモリー使用率が減少
  - 固有値のみ必要な場合、大きな次元で (S/D)SYRDB および (S/D)SYEV のパフォーマンスが向上

- ScaLAPACK
  - デフォルトの NETLIB 複素数型と MKL 複素数型が混在できるように PBLAS ヘッダーを更新
- DFT:複素数-複素数および実数-複素数の変換を最適化
- 転置:縦長の行列と横長の行列で mkl_?omatcopy ルーチンのパフォーマンスが向上
- DFTI インターフェイスと FFTW ラッパーがスレッドセーフになり、並列領域から MKL DFT を使用する場合 NUMBER_OF_USER_THREADS パラメーターは任意設定に変更

6.2.2 インテル® MKL 11.1 の新機能

- 条件付きの数値再現性:アライメントされていないデータで条件付き数値再現性 (CNR) モードをサポート
- Windows* において、インテル® メニー・インテグレーテッド・コア (インテル® MIC) アーキテクチャー・ベースのインテル® Xeon Phi™ コプロセッサーでコンパイラーによるオフロード支援と自動オフロード・プログラミング・モデルをサポート
- 最新の AMD* システムにおいて CNR=AUTO モードのパフォーマンスが向上
- BLAS:
  - インテル® SSE4.2 以降対応のすべてのインテル® プロセッサーにおいて [S/D]GEMV のパフォーマンスが向上
  - インテル® アドバンスト・ベクトル・エクステンション 2 (インテル® AVX2) において [D/Z]GEMM および倍精度のレベル 3 BLAS 関数を最適化
  - インテル® アドバンスト・ベクトル・エクステンション (インテル® AVX) およびインテル® AVX2 において [Z/C]AXPY および [Z/C]DOT[U/C] を最適化
  - インテル® MIC アーキテクチャーにおいて DTRMM のシーケンシャル・バージョンを最適化
  - インテル® AVX2 において DAXPY をチューニング
- LAPACK:
  - 固有値のみ必要な場合、大きな次元で (S/D)SYRDB および (S/D)SYEV のパフォーマンスが向上
  - M,N<10 のような小さなサイズで xGESVD のパフォーマンスが向上。
- VSL:
  - 平均絶対偏差のサポートとサンプルの追加
  - alpha=1 の場合のワイブル乱数ジェネレーター (RNG) のパフォーマンスが向上
  - 外積および平均絶対偏差の行列において、次数 4 までのローデータおよび中央部の統計的総和をサポート
  - S. Joe および F. Y. Kuo により設計された、最大 21,201 次元まで発生できるソボル QRNG の使用法を示す VSL サンプルを追加
  - インテル® MIC アーキテクチャーにおいて SFMT19937 基本乱数ジェネレーター (BRNG) のパフォーマンスが向上
- DFT:
  - インテル® MIC アーキテクチャーにおいて倍精度の複素数-複素数変換のパフォーマンスが向上
  - インテル® AVX2 において複素数-複素数 DFT を最適化
  - インテル® Xeon® プロセッサー E5 v2 ファミリーにおいて 2 次元の複素数-複素数 DFT を最適化
  - インテル® Xeon® プロセッサー E5 ファミリー (インテル® AVX) およびインテル® AVX2 において GENE アプリケーション固有のワークロードでパフォーマンスが向上
  - DFTI 計算関数のドキュメントのデータレイアウトが向上
  - 大規模な実数-複素数 FFT のスケーリング
- データ・フィッティング:

    - インテル® Xeon® プロセッサーおよびインテル® MIC アーキテクチャーにおいて df?Interpolate1D および df?SearchCells1D 関数のパフォーマンスが向上
    - インテル® MIC アーキテクチャー、インテル® Xeon® プロセッサー X5570、インテル® Xeon® プロセッサー E5-2690 において、線形および 3 次 Hermite/Bessel/Akima スプライン用 df?construct1d 関数のパフォーマンスが向上
- 転置
    - 正方行列でインプレース転置のパフォーマンスが向上
- インストール時間を短縮するためパッケージに含まれるインテル® MKL のサンプルとテストをアーカイブ
- リンクツールおよびリンク・ライン・アドバイザー:Windows* でインテル® MIC アーキテクチャーをサポート

6.3 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 (http://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 Extended Eigensolver の機能は、Feast Eigenvalue Solver 2.0 (http://www.ecs.umass.edu/~polizzi/feast/) をベースにしています。

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。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。

7 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。

7.1 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。

7.1.1 ライブラリーの問題

- 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。

8 著作権と商標について

本資料に掲載されている情報は、インテル製品の概要説明を目的としたものです。本資料は、明示されているか否かにかかわらず、また禁反言によるとよらずにかかわらず、いかなる知的財産権のライセンスを許諾するものではありません。製品に付属の売買契約書『Intel's Terms and Conditions of Sale』に規定されている場合を除き、インテルはいかなる責任を負うものではなく、またインテル製品の販売や使用に関する明示または黙示の保証 (特定目的への適合性、商品適格性、あらゆる特許権、著作権、その他知的財産権の非侵害性への保証を含む) に関してもいかなる責任も負いません。インテルによる書面での合意がない限り、インテル製品は、その欠陥や故障によって人身事故が発生するようなアプリケーションでの使用を想定した設計は行われていません。

インテル製品は、予告なく仕様や説明が変更されることがあります。機能または命令の一覧で「留保」または「未定義」と記されているものがありますが、その「機能が存在しない」あるいは「性質が留保付である」という状態を設計の前提にしないでください。これらの項目は、インテルが将来のために留保しているものです。インテルが将来これらの項目を定義したことにより、衝突が生じたり互換性が失われたりしても、インテルは一切責任を負いません。この情報は予告なく変更されることがあります。この情報だけに基づいて設計を最終的なものとしないでください。

本書で説明されている製品には、エラッタと呼ばれる設計上の不具合が含まれている可能性があり、公表されている仕様とは異なる動作をする場合があります。現在確認済みのエラッタについては、インテルまでお問い合わせください。

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、
http://www.intel.com/design/literature.htm (英語) 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。

インテル® C++ コンパイラー、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® マス・カーネル・ライブラリー、およびインテル® スレッディング・ビルディング・ブロックは、インテルのエンド・ユーザー・ソフトウェア使用許諾契約書 (EULA) の下で提供されます。

GNU* プロジェクト・デバッガー (GDB) は、General GNU Public License GPL V3 の下で提供されます。

詳細は、製品に含まれるライセンスを確認してください。

Intel、インテル、Intel ロゴ、Itanium、Pentium、Xeon、Intel Xeon Phi、Cilk は、アメリカ合衆国および / またはその他の国における Intel Corporation の商標です。

* その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。