インテル® Visual Fortran 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: 321417-003JA

2012 年 4 月 11 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® Visual Fortran Composer XE 2011 は、以前「インテル® Visual Fortran 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。

Update 10

- インテル® Fortran コンパイラー 12.1.4
- インテル® マス・カーネル・ライブラリー 10.3 Update 10
- 報告された問題の修正

Update 9

- インテル® Fortran コンパイラー 12.1.3
  - `/assume:std_intent_in` オプションを追加
  - 派生型 Co-Array の ALLOCATABLE または POINTER コンポーネントにおける制限
- インテル® マス・カーネル・ライブラリー 10.3 Update 9
- Visual Studio* 2008 のヘルプにアクセスするときの注意事項
- 報告された問題の修正

Update 8

- インテル® Fortran コンパイラー 12.1.2
- Update 6 で追加された QuickWin ライブラリー・ルーチン GETLINEWIDTHQQ および SETLINEWIDTHQQ の説明をリリースノートに追加
- 試験的並列ビルドのサポートを追加
- インテル® マス・カーネル・ライブラリー 10.3 Update 8
- 報告された問題の修正

Update 7

- インテル® Fortran コンパイラー 12.1.1
- インテル® マス・カーネル・ライブラリー 10.3 Update 7
  - インテル® MKL の注意事項を更新
- Visual Studio* 2010 Shell のインストールには Microsoft* .NET 4.0 Framework が必要
- 報告された問題の修正

Update 6

- Microsoft* Visual Studio* 2010 Shell ベースの統合開発環境を提供以前のバージョンのインテル® Visual Fortran に付属する Visual Studio* 2008 Shell も引き続きサポート
- 新しいトップレベル・フォルダーへの製品のインストール
- インテル® Fortran コンパイラー 12.1.0
  - Windows* 上の分散メモリー環境で Co-Array 機能をサポート
  - SOURCE= を使用した ALLOCATE のサポート
  - Visual Studio* 2010 で多数の新しいエディター機能を追加
  - OpenMP* 3.1 のサポート
  - コンパイラーの主要ドキュメントであるユーザー・リファレンス・ガイドの簡略化と再編成。主な変更点: インテル® コンパイラーの主要機能をまとめた新しいセクション「主な機能」の追加と、コンパイラー・オプション・リファレンスの機能別のグループ化。

    - 『インテル® Visual Fortran を使用した Windows* ベースのアプリケーションの作成とビルド』をインテルの Web サイトに移動
    - インテル® Parallel Debugger Extension のサポート終了予定
- インテル® マス・カーネル・ライブラリー 10.3 Update 6
- 報告された問題の修正

Update 5

- Microsoft* Visual Studio* 2005 のサポート終了予定
- Microsoft* Windows Server* 2003 および Microsoft* Windows Vista* のサポート終了予定
- インテル® Fortran コンパイラー 12.0.5
- インテル® マス・カーネル・ライブラリー 10.3 Update 5
- 報告された問題の修正

Update 4

- インテル® Fortran コンパイラー 12.0.4
- インテル® マス・カーネル・ライブラリー 10.3 Update 4
- 報告された問題の修正

Update 3

- インテル® Fortran コンパイラー 12.0.3
    - /Qsox オプションのデフォルト動作の変更、および含める情報を指定するためのオプションのキーワードの追加
    - 日本語のドキュメントと診断メッセージ
- インテル® マス・カーネル・ライブラリー 10.3 Update 3
- 報告された問題の修正

Update 2

- インテル® Fortran コンパイラー 12.0.2
- インテル® マス・カーネル・ライブラリー 10.3 Update 2
- スタティック・セキュリティー解析でのデータファイルの保存方法の変更
- 報告された問題の修正

Update 1

- インテル® Fortran コンパイラー 12.0.1
- インテル® マス・カーネル・ライブラリー 10.3 Update 1
- 報告された問題の修正

製品リリース (12.0.0)

- 最初の製品リリース

1.2 製品の内容

インテル® Fortran 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® Visual Fortran コンパイラー XE 12.1.4。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® マス・カーネル・ライブラリー 10.3 Update 10
- Microsoft* 開発環境への統合
- Microsoft* Visual Studio* 用インテル® Parallel Debugger Extension 12.1.4
- Microsoft* Visual Studio* 2010 Shell とライブラリー (学生ライセンス、評価版ライセンスでは提供されません。)
- サンプルプログラム
- 各種ドキュメント

インテル® Visual Fortran Composer XE 2011 SP1 Windows* 版 IMSL* 6.0.0 同梱には、上記のほか、Visual Numerics* 社の IMSL Fortran 数値計算ライブラリー 6.0 が含まれています。

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)
- 2GB 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 SP3、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* 2010、Microsoft* Visual Studio* 2010 Shell、Microsoft* Visual Studio* 2008 SP1、または Visual Studio* 2008 SP1 アップデートが適用された Visual Studio* 2008 Shell 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 以降 (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* 2005 Standard Edition 以降 (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* 2010 Shell (インテル® Fortran コンパイラーの特定のライセンスに付属) ベースのインテル® Visual Fortran 開発環境 [2]
    - Microsoft* Visual Studio* 2008 Shell (インテル® Fortran コンパイラー 11.0、11.1、12.0 に付属) ベースのインテル® Visual Fortran 開発環境
- 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 [2]
    - 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 Vista* 用 Microsoft* Windows Software Development Kit Update
    - Windows Server* 2008 および .NET Framework 3.5 用 Microsoft* Windows SDK
- Microsoft* Visual Studio* 2010 Shell には次の制限があります。
    - Windows* XP 64 ビットではサポートされていません。以前のバージョンのインテル® Visual Fortran に付属の Microsoft* Visual Studio* 2008 Shell を使用できます。

    - Microsoft* Visual Studio* 2010 Shell をインストールする前に Microsoft* .NET 4.0 Framework をインストールする必要があります。Windows* XP、Windows Server* 2003 では、インテル® Composer XE 2011 をインストールする前に Microsoft* Windows Imaging Component をインストールする必要があります。詳細は、このリリースノートの「インストール」を参照してください。
- 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、すべてのエディションでこのコンポーネントがデフォルトでインストールされます。
2. Microsoft* Visual Studio* 2010 Shell ベースのインテル® Visual Fortran 開発環境は、インテル® Visual Fortran Composer XE のアカデミック・ライセンスと商用ライセンスに含まれています。学生ライセンス、評価版ライセンスには含まれていません。この開発環境は、Fortran アプリケーションの編集、ビルド、デバッグに必要なものがすべて揃っています。ただし、次のような、Visual Studio* 製品の一部の機能は含まれていません。
    - リソースエディター (代用としてサードパーティー・ツールの ResEdit* (http://www.resedit.net/ (英語)) を参照してください。)
    - Compaq* Visual Fortran プロジェクトの自動変換
    - Visual C++* や Visual Basic* などの Microsoft* の言語ツール
3. Microsoft* Visual C++* 2010 Express Edition とインテル® Visual Fortran Composer XE 2011 に付属の Microsoft* Visual Studio* 2010 Shell は共存できます。C++ と Fortran 開発環境はそれぞれ独立しています。
4. インテル® Visual Fortran 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。
5. 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。

1.3.2 Microsoft* Visual Studio* 2005 のサポート終了予定

インテル® Visual Fortran Composer XE の将来のメジャーリリースでは、Visual Studio* 2005 はサポートされなくなる予定です。Visual Studio* 2005 を使用している場合は、インテルでは Visual Studio* 2010 への移行を推奨しています。

1.3.3 Microsoft* Windows Server* 2003 および Microsoft* Windows Vista* のサポート終了予定

インテル® Visual Fortran Composer XE の将来のメジャーリリースでは、Microsoft* Windows Server* 2003 および Microsoft* Windows Vista* はサポートされなくなる予定です。

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。

1.4.1 『インテル® Visual Fortran を使用したWindows* ベースのアプリケーションの作成とビルド』は Web から入手可能

コンパイラー・ドキュメントの「インテル® Visual Fortran を使用した Windows* ベースのアプリケーションの作成とビルド」は、Intel® Software Documentation Library Web サイトに移動しました。http://intel.ly/WinApp (英語) から PDF 形式のドキュメントをダウンロードできます。

1.4.2 Visual Studio* 2008 のインテル® Composer XE ヘルプに最初にアクセスしたときの遅延

Microsoft* Visual Studio* 2008 にインストールされたヘルプ・ドキュメントに最初にアクセスしたとき、表示に時間がかかる場合があります。これは、Visual Studio* でヘルプを表示する前に、新しいヘルプをコレクションに統合して、コレクションの索引を再生成するためです。Visual Studio* にすでに統合されているヘルプとインストールするヘルプのサイズに応じて、新しくヘルプが表示されるまで数分以上かかる場合があります。

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、日本語と英語の両方を備えたインストーラーで日本語をサポートしています。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。

日本語サポートはすべての製品アップデートで提供されているわけではありません。

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/pla2A5 (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 インストールに必要なソフトウェア

製品に付属する Microsoft* Visual Studio* 2010 Shell をインストールする場合、インテル® Visual Fortran Composer XE 2011 をインストールする前に、追加で Microsoft* ソフトウェアのインストールが必要な場合があります。Microsoft* Visual Studio* 2010 Shell は、Windows* XP 64 ビットではサポートされていません。

Microsoft* Visual Studio* 2010 Shell のインストールには Microsoft* .NET 4.0 Framework が必要です。Microsoft* .NET 4.0 Framework は、次のリンクからインストーラーをダウンロードできます。

- .NET 4.0 Framework 32 ビットおよび 64 ビット

Windows* 7 または Windows Server* 2008 にインストールする場合は、Shell のインストール時に .NET Framework 4.0 が (システムにインストールされていない場合) 自動的にダウンロードされインストールされます。この処理に失敗すると、エラーメッセージが表示され、インテル® Visual Fortran Composer はインストールされません。Shell のインストールに失敗した場合は、上記のリンクから .NET 4.0 Framework をダウンロードしてやり直してください。

Windows* XP または Windows Server* 2003 にインストールする場合は、Microsoft* Windows Imaging Component も必要です。Microsoft* Windows Imaging Component は、次のリンクからインストーラーをダウンロードできます。

- Windows Imaging Component 32 ビット
- Windows Imaging Component 64 ビット

DVD または Visual Studio* 2010 Shell 付属製品のダウンロード版を使用してインテル® Visual Fortran Composer XE 2011 をインストールする場合、マシンに Visual Studio* 2010 がインストールされていないと、インストーラーは Visual Studio* 2010 Shell をインストールしようとします。Visual Studio* 2010 Shell をインストールしない場合は、「カスタム・インストール」を選択して、Visual Studio* 2010 Shell のチェックをオフにするか、"_novsshell.exe" (Visual Studio* 2010 Shell なし) のインストーラーをダウンロードして使用してください。

2.1.2 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* 2008 Shell 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。

Microsoft* Visual Studio* 2010 を使用している場合、このステップは必要ありません。

2.1.3 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 ページからリンクを提供) 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。また、「Microsoft* Visual Studio* 2005 の管理者権限に関するメッセージダイアログ」も参照してください。

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* エクスプローラーで 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。以前のバージョンの削除は、このバージョンをインストールする前でも後でも行うことができます。

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 でシリアル番号を登録すると、製品のアップデートや以前のバージョンを利用できます。

2.2.1 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。

2.2.2 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、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 SP1\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。

2.2.3 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/oPEdEe (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 FLEXlm* ライセンス・マネージャーに関する情報も記述されています。

2.2.4 IMSL Fortran 数値計算ライブラリーのインストール

インテル® Visual Fortran Composer XE IMSL 6.0 同梱版のライセンスをお持ちの場合、コンパイラーとは別に IMSL をインストール (ダウンロードまたはディスクのいずれか) する必要があります。IMSL ライブラリーをインストールする前にコンパイラーをインストールしてください。

2.2.5 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® Visual Fortran 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® Visual Fortran 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® Visual Fortran 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® Visual Fortran 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/en-us/library/dd264831.aspx を参照するか、microsoft.com で「ヘルプ ライブラリ マネージャー」を検索してください。

2.2.6 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*および Windows* 7 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] / [プログラムと機能] 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。インストールした製品に応じて、以下のいずれかのエントリーが表示されます。

- インテル® Visual Fortran Composer XE 2011 Windows* 版
- インテル® Composer XE 2011 Windows* 版
- インテル® Parallel Studio XE 2011 SP1 Windows* 版

IMSL* ライブラリーをインストールした場合は、以下のエントリーも表示されます。

- インテル(R) Visual Fortran Composer XE 2011 用 IMSL* Fortran ライブラリー 6.0

コンパイラーのインストールの一部として Microsoft* Visual Studio* 2010 Shell をインストールした場合、以下の追加エントリーが表示されます。

- Microsoft* Visual Studio* 2010 Shell (Integrated) - JPN
- インテル(R) Visual Fortran 用 Microsoft* Visual Studio* 2010 ファイル

製品を完全に削除する場合を除き、これらのエントリーは削除しないでください。

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。アップデートを最初にインストールする場合、古いバージョンを置換するか、システムで古いバージョンと新しいバージョンの両方を使用するかを選択します。この選択は、将来のアップデートにも適用されます。Microsoft* Visual Studio* の [ツール] > [オプション] > [インテル(R) Visual Fortran] > [コンパイラー] から、使用するコンパイラーのバージョンを選択できます。11.1 よりも古いコンパイラーは、Visual Studio* 2005 または 2008 からは選択できません。また、12.0 よりも古いコンパイラーは、Visual Studio* 2010 からは選択できません。インストールされているすべてのバージョンをコマンドラインから使用できます。

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 サイレント・インストール/アンインストール

コンパイラーの自動インストール/アンインストールについては、http://intel.ly/qAwdvR (英語) を参照してください。

2.5 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。Update 6 では、トップレベルのフォルダーが変更されています。システム環境変数 IFORT_COMPILER12 を使用して、マシンにインストールされている最新バージョンのインテル® Visual Fortran Composer XE 2011 を検出できます。

- C:\Program Files\Intel\Composer XE 2011 SP1
  - bin
    - ia32
    - ia32_intel64
    - intel64
  - compiler
    - include
      - ia32
      - intel64
    - lib
      - ia32
      - intel64
  - debugger
  - Documentation
  - help
  - mkl
    - benchmarks
    - bin
    - examples
    - include
    - interfaces
    - lib
    - tests
    - tools
  - redist
  - Samples

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 2011 SP1.nnn (nnn はアップデート番号) に変更されます。

2.6 インストールの既知の問題

2.6.1 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® Visual Fortran 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® Visual Fortran 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® Visual Fortran コンパイラー

このセクションでは、インテル® Visual Fortran 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

一般に、インテル® Fortran コンパイラーの以前のバージョン (8.0 以降) でコンパイルされたオブジェクト・コードおよびモジュールは、バージョン 12.0 でもそのまま使用できます。ただし、次の例外があります。

- バージョン 12.0 よりも前のコンパイラーを使用してビルドされた CLASS キーワードを使用して多相変数を宣言しているソースは再コンパイルする必要があります。
- マルチファイルのプロシージャー間の最適化 (/Qipo) オプションを使用してビルドされたオブジェクトは再コンパイルする必要があります。
- バージョン 12.0 よりも前のコンパイラーを使用してビルドされた REAL(16)、REAL*16、COMPLEX(16)、COMPLEX*32 データ型を使用しているオブジェクトは再コンパイルする必要があります。
- バージョン 10.0 よりも前のコンパイラーを使用してインテル® 64 アーキテクチャー用にビルドされたモジュール変数を含むオブジェクトは再コンパイルする必要があります。Fortran 以外のソースからこれらの変数を参照する場合、不正な先頭の下線を削除するように外部名を変更する必要があります。
- バージョン 11.0 よりも前のコンパイラーを使用してコンパイルされた、ATTRIBUTES ALIGN 宣言子を指定したモジュールは再コンパイルする必要があります。この問題が発生した場合、問題を通知するメッセージが表示されます。

3.1.1 REAL(16) および COMPLEX(16) データ型のスタック・アライメントの変更 (12.0)

以前のリリースでは、REAL(16) または COMPLEX(16) (REAL*16 または COMPLEX*32) 項目が値で渡されたとき、スタックアドレスは 4 バイトでアラインされていました。パフォーマ

ンスを向上させるため、バージョン 12 では、コンパイラーはこれらの項目を 16 バイトでアラインします。引数は 16 バイト境界でアラインされます。

この変更は、主にライブラリーが生成した REAL(16) 値の計算を行うライブラリー (組込み関数を含む) の呼び出しに影響します。以前のバージョンでコンパイルしたコードをバージョン 12 のライブラリーとリンクする場合、またはアプリケーションをインテルのランタイム・ライブラリーの共有バージョンにリンクする場合、正しくない結果が返される可能性があります。

この問題を回避するには、REAL(16) および COMPLEX(16) データ型を使用しているすべての Fortran ソースを再コンパイルしてください。

3.2 新規および変更されたコンパイラー機能

一部の言語機能に関する説明はコンパイラーのドキュメントにはまだ含まれていません。必要に応じて、Fortran 2003 規格 (http://j3-fortran.org/doc/2003_Committee_Draft/04-007.pdf) および Fortran 2008 規格 (http://j3-fortran.org/doc/standing/links/007.pdf) を参照してください。

3.2.1 Fortran 2003 の機能

- FINAL サブルーチン
- 型バインド・プロシージャーの GENERIC キーワード
- 汎用インターフェイスの名前は派生型と同じ名前を使用可能
- ポインター代入の境界の仕様と境界の再マップリスト
- SOURCE= を使用した ALLOCATE (Update 6)

3.2.2 Fortran 2008 の機能

- 配列の最大次元数が 31 次元に (Fortran 2008 では 15 次元)
- Co-Array
    - CODIMENSION 属性
    - SYNC ALL 文
    - SYNC IMAGES 文
    - SYNC MEMORY 文
    - CRITICAL および END CRITICAL 文
    - LOCK および UNLOCK 文
    - ERROR STOP 文
    - ALLOCATE および DEALLOCATE で Co-Array を指定
    - 組込みプロシージャー: IMAGE_INDEX、LCOBOUND、NUM_IMAGES、THIS_IMAGE、UCOBOUND
    - **注:** ATOMIC_DEFINE および ATOMIC_REF はサポートしていません
- CONTIGUOUS 属性
- ALLOCATE の MOLD キーワード
- DO CONCURRENT
- OPEN の NEWUNIT キーワード
- G0 および G0.d フォーマット編集記述子
- 無制限のフォーマット項目繰り返しカウント指定子
- CONTAINS セクションは空にすることも可能
- 組込みプロシージャー: BESSEL_J0、BESSEL_J1、BESSEL_JN、BESSEL_YN、BGE、BGT、BLE, BLT、DSHIFTL、DSHIFTR、ERF、ERFC、ERFC_SCALED、GAMMA、HYPOT、IALL、IANY、IPARITY、IS_CONTIGUOUS、LEADZ、LOG_GAMMA、MASKL、MASKR、MERGE_BITS、NORM2、PARITY、POPCNT、POPPAR、SHIFTA、SHIFTL、SHIFTR、STORAGE_SIZE、TRAILZ
- 組込みモジュール ISO_FORTRAN_ENV の追加: ATOMIC_INT_KIND、ATOMIC_LOGICAL_KIND、CHARACTER_KINDS、INTEGER_KINDS、INT8、INT16、INT32、

INT64、LOCK_TYPE、LOGICAL_KINDS、REAL_KINDS、REAL32、REAL64、REAL128、STAT_LOCKED、STAT_LOCKED_OTHER_IMAGE、STAT_UNLOCKED
- (Update 6) ALLOCATABLE または POINTER 属性を持たない OPTIONAL 仮引数は、対応する実引数に ALLOCATABLE 属性があるのに割り当てられない場合、POINTER 属性があるのに関連付けが解除されている場合、または組込み関数 NULL への参照の場合、無視されます。
- (Update 6) 仮引数がプロシージャー・ポインターの場合、そのポインターの有効な参照先か、または組込み関数 NULL への参照である実引数に関連付けられます。実引数がポインターではない場合、仮引数に INTENT (IN) 属性が含まれていなければなりません。

3.2.3 Co-Array

共有メモリー環境で Co-Array を使用するプログラムの実行に特別なプロシージャーは必要ありません。実行ファイルを実行するだけでかまいません。根本的な並列化の実装にはインテル® MPI が使用されます。コンパイラーをインストールすると、共有メモリーでの実行に必要なインテル® MPI ランタイム・ライブラリーが自動的にインストールされます。

Update 6 では、Windows* クラスター上の分散メモリーを使用して実行する Co-Array アプリケーションがサポートされるようになりました。`/coarray:distributed` オプションを使用するには、インテル® クラスター・ツールキットのライセンスが必要です。Windows* 上で分散 Co-Array アプリケーションを実行する方法については、http://intel.ly/oZurZS (英語) を参照してください。

現在、インテル® MPI 以外の MPI 実装や OpenMP* を使用した Co-Array アプリケーションの使用はサポートされていません。

デフォルトでは、作成されるイメージの数は現在のシステムの実行ユニットの数と同じです。メインプログラムをコンパイルする ifort コマンドで `/Qcoarray-num-images:<n>` オプションを指定することで、この設定を変更することができます。また、環境変数 `FOR_COARRAY_NUM_IMAGES` でイメージ数を指定することもできます。

3.2.3.1 Co-Array の既知の問題

このバージョンでは、以下の機能は動作しません。

- 別のイメージを参照している Co-Array の配列スライスの出力 (WRITE、PRINT など)。配列全体の参照または単一要素は機能します。
- REAL(16) または COMPLEX(16) の Co-Array のデフォルト初期化
- 派生型 Co-Array の ALLOCATABLE または POINTER コンポーネントの別のイメージの値へのアクセス

3.2.4 新しい宣言子と追加された宣言子 (Update 6)

インテル® Composer XE 2011 Update 6 では、次のコンパイラー宣言子が追加、変更されています。詳細は、ドキュメントを参照してください。

- ATTRIBUTES VECTOR
- NOFUSION
- PARALLEL 宣言子で FIRSTPRIVATE 節を指定可能
- SIMD 宣言子で FIRSTPRIVATE 節または LASTPRIVATE 節を指定可能

3.2.5 OpenMP* の変更 (Update 6)

インテル® Composer XE 2011 Update 6 では、OpenMP* サポートに関して次の点が変更されています。

- OpenMP* 3.1 のサポート
- TASKYIELD 宣言子
- ATOMIC 宣言子に新しい節を追加

- TASK 宣言子で FINAL 節および MERGEABLE 節を指定可能

3.2.6 新しい QuickWin ライブラリー・ルーチン (Update 6)

インテル® Visual Fortran Composer XE 2011 Update 6 では次の新しい QuickWin ライブラリー・ルーチンが追加されましたが、製品のドキュメントには含まれていません。

3.2.6.1 GETLINEWIDTHQQ

グラフィック関数: 現在の線の幅または最後の SETLINEWIDTHQQ の呼び出しで設定された線の幅を取得します。

モジュール: USE IFQWIN

result = GETLINEWIDTHQQ()

結果

結果は INTEGER(4) です。線の現在の幅がピクセル単位で取得されます。

プログラム (または特定のグラフィック・ウィンドウ) で SETLINEWIDTHQQ の呼び出しが行われていない場合は 1 を返します(グラフィック・ルーチンで描画される線のデフォルトの幅は 1 ピクセルです。)

3.2.6.2 SETLINEWIDTHQQ

グラフィック・サブルーチン: サポートしているグラフィック関数を使用して描画された実線の幅を設定します。

モジュール: USE IFQWIN

CALL SETLINEWIDTHQQ (x)

x　　(Input) INTEGER(4)。負ではない整数を指定します。

このサブルーチンは、引数として渡された値を使用して線の幅をピクセル単位で設定します。

SETLINEWIDTHQQ は、LINETO、POLYGON、LINETOAR、LINETOAREX、RECTANGLE などの関数を使用した直線の描画に影響を与えます。また、ARC、ELLIPSE、PIE などの関数を使用した曲線の描画にも影響を与えます。

注

Windows* API CreatePen() の nWidth 引数は、線または閉じた図形の境界線を描画するために使用される幅です。コスメティック・ペンでは、1 ピクセルの幅のみ指定できます。1 より大きな幅を指定した場合は無視されます。ジオメトリック・ペンには 1 ピクセルより大きな幅を指定できますが、線は実線または空にしかできません。PS_DASH、PS_DOT、PS_DASHDOT、PS_DASHDOTDOT をスタイルとして指定し、1 ピクセルよりも大きな幅を指定した場合、線は PS_SOLID として描画されます。

関連情報

Microsoft* MSDN ドキュメントの Windows* API CreatePen。

3.2.7 その他の変更

- 識別子によるクロスリファレンス付きのソース・リスト・ファイルを作成するための機能の追加
- ガイド付き自動並列化
- 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
- コンパイラーのデフォルトと Fortran 2003 のセマンティクスが一致しないすべての構文で Fortran 2003 セマンティクスを仮定するように指定するためのオプション

- ビルドの依存関係をファイルに出力するための機能の追加
- Visual Studio* プロジェクトにおいて、サブプロジェクトのモジュール出力ディレクトリーを親プロジェクトの “追加 INCLUDE ディレクトリー” として自動で追加

3.3 新規および変更されたコンパイラー・オプション

詳細は、コンパイラーのドキュメントを参照してください。

3.3.1 インテル® Composer XE 2011 の新規および変更されたコンパイラー・オプション (Update 6 以降)

- /align [no]qcommons
- /arch:CORE-AVX-I
- /arch:CORE-AVX2
- /assume:[no]std_intent_in (Update 9)
- /openmp
- /QaxCORE-AVX-I
- /QaxCORE-AVX2
- /Qfma[-]
- /Qopt-mem-layout-trans[:n]
- /QxCORE-AVX-I
- /QxCORE-AVX2
- /QxSSSE3_ATOM

3.3.2 インテル® Composer XE 2011 の新規および変更されたコンパイラー・オプション (Update 5 まで)

- /assume:[no]fpe_summary
- /assume:old_ldout_format
- /gen-dep
- /gen-depformat
- /list
- /list-line-len
- /list-page-len
- /Qcoarray (上記を参照)
- /Qcoarray-num-images
- /Qcov-dir
- /Qcov-file
- /Qcov-gen
- /Qdiag-sc-dir
- /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
- /Qopt-args-in-regs
- /Qopt-matmul
- /Qpar-runtime-control

- /Qpatchable-addresses
- /Qprof-value-profiling
- /Qprofile-functions
- /Qprofile-loops
- /Qprofile-loops-report
- /Qsimd
- /Qsox=keyword
- /Qzero-initialized-in-bss
- /show
- /standard-semantics

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。

3.3.3 /assume:[no]std_intent_in の追加 (Update 9)

`/assume:[no]std_intent_in` オプションは、Fortran 規格に従って INTENT(IN) 属性の仮引数が呼び出し間で変更されないことをコンパイラーが仮定するかどうかを決定します。デフォルトは `std_intent_in` で、コンパイラーは INTENT(IN) 引数が変更されないことを仮定します。`nostd_intent_in` の場合は仮定しません。`/standard-semantics` オプションを指定すると、`/assume:std_intent_in` オプションが指定されます。

3.3.4 /Qsox オプションの追加キーワード、デフォルトの変更 (Update 3)

オブジェクト・ファイルおよび実行ファイルに使用されたコンパイラー・オプションとプロシージャーのプロファイル情報を追加するための `/Qsox` オプションは、インライン展開された関数のリストを含めたり、プロシージャーのプロファイル情報を除外できるようになりました。

`/Qsox` の構文は次のように変更されました。

```
/Qsox[-]
/Qsox=keyword[,keyword]
```

`keyword` には、`inline` または `profile` のいずれかを指定できます。キーワードなしで `/Qsox` を指定すると、以前のリリースとは異なり、コマンドライン・オプションの情報のみが追加されます。以前のリリースと同じ動作にするには、`/Qsox=profile` を使用してください。`/Qsox` オプションはコマンドラインで複数回指定することができますが、その場合は左から右の順に解釈されます。

この情報は、オブジェクト・ファイルにコメントとして追加されます。Visual Studio* 2005 以降の Microsoft* のリンカーではこれらのコメントは無視されるため、実行ファイルにはこの情報は含まれません。

3.4 試験的並列ビルドオプション (Update 8)

Visual Studio* ビルド環境に、マルチコアまたはマルチプロセッサー・システムで未解決の依存性がないソースを並列ビルドできる機能が追加されました。この機能を利用すると、大規模なプロジェクトのビルドに必要な時間を短縮できます。

この機能を有効にするには、プロジェクトのプロパティーで [Fortran] > [Command Line (コマンドライン)] を開きます。[Additional Options (追加のオプション)] フィールドで、最初の (または唯一の) オプションとして /MP を追加します。/MP が行の最初のオプションでない場合、並列ビルドは行われません。

これは最初の実装です。オプションの設定方法やファイルのビルド順は将来のバージョンで変更される場合があります。

3.5 Visual Studio* 2010 のソースエディターの拡張 (Update 6)

インテル® Visual Fortran Composer XE 2011 Update 6 では、Microsoft* Visual Studio* 2010 での Fortran ソースエディターの新機能が追加されました。これら機能は、Visual Studio* 2010 よりも前のバージョンでは利用できません。いくつかの新機能は、デフォルトでは無効になっています。これらの機能は、Visual Studio* 2010 の [Tools (ツール)] > [Options (オプション)] > [Text Editor (テキスト エディター)] > [Fortran] > [Advanced (詳細)] から有効/無効にできます。このダイアログにある [Disable Database (データベースの無効化)] は、デフォルトでは False (つまり、データベースは有効) に設定されています。このオプションを True (データベースを無効) に設定すると、次のいくつかの機能も無効になります。

3.5.1 ナビゲーション・バー

ナビゲーション・バーは、[ソースエディター] ペインの上に表示されます。2 つのセクションから成り、左側では現在のソースファイルで定義されているモジュールを選択し、右側では現在のソースで定義されているすべてのプロシージャーの一覧からプロシージャーをクリックしてジャンプできます。

3.5.2 スマートインデント

エディターによりブロック構造 (IF、DO など) と対応する END 文のインデントが自動的に調節されます。この機能の動作設定は、[Tools (ツール)] > [Options (オプション)] > [Text Editor (テキスト エディター)] > [Fortran] > [Tabs (タブ)] から調整できます。

3.5.3 コード・アウトライン

オプション: Enable Outlining (アウトライン・モードを有効にする)、Outline Statement Blocks (ステートメント・ブロックのアウトライン化)
デフォルト: [Enable Outlining (アウトライン・モードを有効にする)] は True、[Outline Statement Blocks (ステートメント・ブロックのアウトライン化)] は False

[Enable Outlining (アウトライン・モードを有効にする)] のみが True の場合、エディター内で PROGRAM、SUBROUTINE、FUNCTION、MODULE、または BLOCK DATA の横にある "- " 記号をクリックしてプログラム単位全体を折りたたむことができます。プログラム単位のコンテンツが折りたたまれ、代わりに "..." 記号が表示され、ボタンが "+" に変更されます。"+" ボタンをクリックすると展開できます。

[Outline Statement Blocks (ステートメント・ブロックのアウトライン化)] も True の場合、IF や DO などのブロック構造も折りたたむことができます。

3.5.4 ブロック区切り記号マッチング

オプション: Highlight Matching Tokens (一致するトークンの強調表示)
デフォルト: False

[Highlight Matching Tokens (一致するトークンの強調表示)] が True の場合、ブロック構造の開始または終了キーワード (IF/THEN/ELSE/END IF、DO/END DO、SELECT/END SELECT など) をクリックして、同じレベルの対応するキーワードを強調表示できます。

このオプションが有効な場合、指定子をクリックすると、指定子へのほかの参照も強調表示されます。

3.5.5 コードスニペット

ソースファイルの任意の場所で右クリックして、[Insert Snippet... (スニペット の挿入...)] を選択すると、DO WHILE や MODULE などのプロトタイプ構造の一覧から任意の項目をダブルクリックして選択することができます。名前を含む構造の場合、一時的な名前が挿入され、選択されます。名前が選択された状態で入力すると、名前が変更され、END 文も一致するように置換されます。

スニペットには「省略名」もあります。省略名を入力して TAB キーを押すとスニペットを挿入できます。

3.5.6 定義へ移動とすべての参照の検索

オプション: Collect Object Browser Information (オブジェクト・ブラウザー情報の収集)、Enable Find All References (すべての参照の検索を有効化)、Enable Go To Definition (定義へ移動を有効化)、Scan System Includes (システム・インクルードをスキャン)
デフォルト: False

上記のオプションすべてが True の場合、宣言された識別子がプロジェクトのソースファイルとシステム・インクルード・ファイルでスキャンされます。識別子を右クリックして、[Go To Definition (定義へ移動)] (デフォルトのショートカット・キーは F12) または [Find All References (すべての参照の検索)] を選択できます。この機能は、「ソースブラウザー」とも呼ばれます。

[Go To Definition (定義 へ移動)] は、識別子が宣言されている場所にカーソルを移動します。必要な場合は、識別子が宣言されているソースファイルを開きます。識別子が IFWINTY などのシステムモジュールで宣言されている場合は、そのモジュールのソースが表示されます。

[Find All References (すべての参照の検索)] は、プロジェクト内にある選択された識別子へのすべての参照の一覧を表示します。参照をクリックすると、その参照に移動できます。

3.5.7 シンボルの検索

オプション: Go To Definition (定義へ移動) と同じ
デフォルト: False

[Edit (編集)] > [Find and Replace (検索と置換)] > ［Find Symbol (シンボルの検索)] (ショットカット・キーは Alt + F12) を選択すると、プロジェクト内にあるシンボルまたはシンボルの部分文字列を検索するためのダイアログが表示されます。

3.5.8 オブジェクト・ブラウザー

オプション: Go To Definition (定義へ移動) と同じ
デフォルト: False

True の場合、[View (表示)] > [Object Browser (オブジェクト ブラウザー)] を選択すると、プロジェクト内のプロシージャーが階層ツリーで表示されます。

3.5.9 組込み関数のクイック情報

オプション: Enable Intrinsic Quick Info (組込み関数クイック情報を有効化)、Enable Intrinsic Parameter Info (組込みパラメーター情報を有効化)
デフォルト: False

これらのオプションが True の場合、マウスポインターを組込みプロシージャー名に移動すると、ツールヒントにその組込み関数の簡単な説明が表示されます。組込みプロシージャー名と (を入力すると、そのプロシージャーの情報と引数が表示されます。

3.5.10 タスクリスト・コメント

オプション: Enumerate Comment Tasks (コメントタスクの列挙)
デフォルト: True

Visual Studio* のタスクリストで収集される特別な書式のコメントを追加できます。タスクコメントは "!" コメント開始文字 (ほかのコメント開始文字はサポートされていません) で始まり、[Tools (ツール)] > [Options (オプション)] > [Environment (環境)] > [Task List (タスク一覧)] で定義されている「トークン」のいずれかが続きます。コメント開始文字の前後にスペースがあっても構いません。次に例を示します。

```
!TODO Make sure this gets filled in
```

これはタスクコメントとして認識されます。Visual Studio* では、HACK、TODO、UNDONE などのいくつかのトークンが事前に定義されています。必要に応じて、このリストは編集できます。トークンには優先順位 (高、標準、低) を指定することも可能です。

プロジェクトのタスクリストを表示するには、[View (表示)] > [Other Windows (その他 のウィンドウ)] > [Task List (タスク一覧)] を選択し、ドロップダウン・ボックスから「コメント」を選択します。項目をダブルクリックすると、そのソース行に移動できます。

3.6 その他の変更

3.6.1 ビルド環境コマンドスクリプトの変更 (12.0)

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。ia32、ia32_intel64、intel64 のいずれかです。vs は任意で指定します。vs2010、vs2008、または vs2005 のいずれかです。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。

注: インストールされている Visual Studio* のバージョンが Visual Studio* 2008 Shell の場合は、vs を vs2008shell と指定するか、省略できます。

また、インテル® C++ 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® C++ 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。

3.6.2 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 (12.0)

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。

3.6.2.1スタティック・セキュリティー解析の動作変更 (Update 2)

インテル® Composer XE 2011 に含まれる inspxe-runsc コマンドライン・ユーティリティーが変更されました。この変更は、インテル® Composer XE 2011 を使用してスタティック・セキュリティー解析 (SSA) を実行する場合にのみ影響します。SSA を使用しない場合や、このユーティリティーを使用せずに SSA を実行する場合には影響ありません。SSA はインテル® Parallel Studio XE 2011 またはインテル® C++ Studio XE 2011 でのみ利用できます。そのため、これらの製品をお使いでない場合は影響ありません。

inspxe-runsc は、アプリケーションのビルド方法を示す**ビルド仕様**を実行します。通常、ビルド仕様ファイルは、ビルドを実行して、実際に行われたコンパイルとリンクを記録することにより生成されます。inspxe-runsc は、インテル® コンパイラーを SSA モードで使用して、再度この処理を行います。SSA 結果はリンクステップで生成されるため、inspxe-runsc で複数のリンクステップを持つビルドが含まれるビルド仕様を実行すると、複数の SSA 結果が生成されます。

インテル® Composer XE 2011 およびインテル® Composer XE 2011 Update 1 の `inspxe-runsc` は、すべての SSA 結果を同じディレクトリーに生成します。リンクが複数ある場合、これは、1 つのプロジェクトの SSA 結果は同じディレクトリーに 1 つだけでなければならないという規則に違反します。新しいバージョンの `inspxe-runsc` は、リンクステップごとの結果を個別のディレクトリーに生成することで、この規則に従っています。ディレクトリー名は、リンクされるファイルの名前を基に付けられます。2 つの実行ファイル file1.exe と file2.exe をビルドするプロジェクトのビルド仕様の場合、以前のバージョンの `inspxe-runsc` では、file1 の結果と file2 の結果 (例えば r000sc と r001sc) が同じディレクトリーに作成されます。新しいバージョンの `inspxe-runsc` でも結果は 2 つ作成されますが、file1 の結果は "My Inspector XE results – file1/r000sc"、file2 の結果は "My Inspector XE results – file2/r000sc" というように別々のディレクトリーに作成されます。2 つの結果のディレクトリーは同じ親ディレクトリーの下に作成されます。

`inspxe-runsc` には、結果の作成場所を指定するための `-result-dir` (`-r`) コマンドライン・スイッチがあります。このスイッチの動作が変更されました。以前は、r000sc のように結果が作成されるディレクトリーの名前を指定していましたが、現在は、"My Inspector XE Results - name" のように結果が作成されるディレクトリーの親ディレクトリーを指定します。つまり、`-r` スイッチのディレクトリー名は、結果の生成される場所から 2 つ上のディレクトリーのものになります。

`inspxe-runsc` のこの変更により、結果ディレクトリーが効率良く移動します。この変更に伴い、ユーザーによる対応が必要になります。`-r` スイッチを指定して `inspxe-runsc` を呼び出すスクリプトを使用している場合は、新しい動作に合わせて、`-r` スイッチの引数を変更してください。また、新しいバージョンの `inspxe-runsc` によって生成される SSA 結果が、以前のバージョンの `inspxe-runsc` によって生成された結果と同じディレクトリーに保存されることがないように、以前の結果ファイルを新しいディレクトリーに移動する必要があります。以前のバージョンの `inspxe-runsc` でリンクステップが 1 つのみのビルド仕様を実行した結果は、"My Inspector XE results – name" という形式のディレクトリーに移動します。この操作を行わないと、新しく作成される結果ですべての問題が "新規" として表示されます。以前のバージョンの inspxe-runsc で複数のリンクステップを含むビルド仕様を実行した場合、SSA ではさまざまな問題がありましたが、これらの問題は新しいバージョンを使用することで解決されます。この場合、以前の結果のうち最も新しいものを "My Inspector XE results – name" という形式の新しいディレクトリーに (1 つのディレクトリーに 1 つの結果が含まれるように) コピーします。これにより、新しいバージョンで作成される結果に以前の問題ステート情報が正しく適用される可能性が高くなります。

3.6.3 OpenMP* レガシー・ライブラリーの削除

OpenMP* のレガシー・ライブラリーはバージョン 12.0 で削除されました。"互換性がある" ライブラリーのみ提供されます。

3.6.4 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。

3.6.5 RANF 移植関数の組込み関数への変更

移植ライブラリーの RANF 関数は非標準の乱数ジェネレーターです。コンパイラー 12.0 では、RANF は新しいハイパフォーマンスな組込み関数として実装されています。プログラムで USE IFPORT を使用して RANF にアクセスしている場合、変更はありません。古いバージョンが使用されます。プログラムで USE IFPORT を使用していない場合、または INTRINSIC RANF を使用している場合、古いバージョンとは異なるシーケンスを返す新しいバージョンが使用されます。RANF のシードはこれまでどおり移植サブルーチン SRAND に

よって設定されます。インテルは、標準の組込み関数 RANDOM_NUMBER の使用を推奨していますが、既存のアプリケーションとの互換性を確保するために RANF も提供しています。

3.6.6 インテル® Parallel Debugger Extension のサポート終了予定

将来のメジャーリリースでは、インテル® Parallel Debugger Extension は削除される可能性があります。削除された場合、次の機能が使用できなくなります。

- OpenMP* ウィンドウ
- データ共有検出 (代わりに、インテル® Inspector XE のデータ共有検出機能を利用できます)
- ベクトルレジスター・ウィンドウ (Visual Studio* の標準のレジスターウィンドウでもベクトルレジスターを確認できます)

3.7 既知の問題

3.7.1 Co-Array の問題

Fortran 2008 Co-Array サポートの既知の問題の一覧は、「Co-Array の既知の問題」を参照してください。

3.7.2 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。

3.7.3 割り当て可能な配列と OpenMP* の PRIVATE/FIRSTPRIVATE

コンパイラーは、OpenMP* の PRIVATE 指示節または FIRSTPRIVATE 指示節で定義された割り当て可能な配列を正しく初期化できません。この問題は将来のアップデートで修正される予定です。これらを組み合わせたときに問題が発生する場合は、`/switch:omp3_private` オプションを追加してください。ただし、これは一時的な回避方法です。この問題の Intel issue ID は DPD200160978 です。

3.8 Microsoft* Visual Studio* 2010 に関する注意事項

Microsoft* Visual Studio* 2010 によりいくつかの変更があります。そのほとんどは、メインプログラムが C/C++ の言語が混在したアプリケーションのビルドに影響するものです。

3.8.1 インテル® Fortran ランタイム・ライブラリーを参照するための Microsoft* Visual C++* の設定

以前のリリースでは、インテル® Fortran の LIB フォルダーを C/C++ プロジェクトで利用できるようにするために [ツール] > [オプション] > [プロジェクトおよびソリューション] > [Visual C++ ディレクトリ] で設定を行っていました。Visual Studio* 2010 では、この方法が変更されています。

1. Visual Studio* で C++ プロジェクトを含むソリューションを開き、[表示] > [プロパティ マネージャー] を選択します。[表示] メニューの直下に [プロパティ マネージャー] が見つからない場合は、[表示] > [その他のウィンドウ] の下にあります。[プロパティ マネージャー] ダイアログボックスが表示されます。これは、[プロパティ] ウィンドウや [プロパティ ページ] とは関係ありません。
2. プロパティー・ツリーの Debug | Win32 の横にある三角または + 記号をクリックしてこのフォルダーを展開します。
3. Microsoft.Cpp.Win32.user をダブルクリックします。
4. [VC++ ディレクトリ] を選択します。

5. [ライブラリ ディレクトリ] の右側のフィールドをクリックします。
6. ドロップダウンから <編集...> を選択します。
7. [新しい行] ボタンをクリックするか、Ctrl+Insert キーを押します。
8. 表示された新しいフィールドに、次のように入力します。

   ```
   $(IFORT_COMPILER12)\compiler\lib\ia32
   ```
9. [OK] をクリックします。もう一度 [OK] をクリックして、[プロパティ ページ] も閉じます。
10. Visual Studio* のメニューから [ファイル] > [すべてを保存] を選択します。

インテル® 64 (x64) 構成でビルドする場合は、次の手順を実行してください。

1. [プロパティ マネージャー] を開いて、Debug | x64 フォルダーを展開します。
2. Microsoft.Cpp.x64.user をダブルクリックします。
3. [VC++ ディレクトリ] を選択します。
4. [ライブラリ ディレクトリ] の右側のフィールドをクリックします。
5. ドロップダウンから <編集...> を選択します。
6. [新しい行] ボタンをクリックするか、Ctrl+Insert キーを押します。
7. 表示された新しいフィールドに、次のように入力します。

   ```
   $(IFORT_COMPILER12)\compiler\lib\intel64
   ```
8. [OK] をクリックします。もう一度 [OK] をクリックして、[プロパティ ページ] も閉じます。
9. Visual Studio* のメニューから [ファイル] > [すべてを保存] を選択します。

[ソリューション エクスプローラー] タブをクリックするか、Ctrl+Alt+L キーを押して [ソリューション エクスプローラー] を表示します。

Debug | x64 フォルダーに Microsoft.Cpp.x64.user プロパティー・ページが見つからない場合は、フォルダーを右クリックして [新しいプロジェクト プロパティ シートの追加] を選択します。そして、MsBuild 4.0 プロパティー・ページの場所を参照します。Windows* XP では、通常以下の場所にあります。

```
C:\Documents and Settings\<username>\Local Settings\Application Data
\Microsoft\MSBuild\v4.0
```

Windows Vista* および Windows* 7 では、通常以下の場所にあります。

```
C:\Users\<username>\Local Settings\AppData\Local\Microsoft\MSBuild\v
4.0
```

これらのパスを表示するためには、隠しファイルと隠しフォルダーの表示を有効にする必要があります。

Microsoft.Cpp.x64.user.props を選択して [開く] をクリックします。後は、上記の手順に従ってください。

3.8.2 プロジェクトの依存関係の調整

以前のバージョンの Visual Studio* から依存関係が設定されているプロジェクトを変換する場合、既存のプロジェクトの依存関係は Visual Studio* 2010 によって参照に変換されます。C/C++ プロジェクトで Fortran プロジェクトを参照している場合、C/C++ プロジェクトのビルドで MSB4075 エラーが発生することがあります。この問題を解決するには、次の操作を行います。

1. C/C++ プロジェクトを右クリックして、[参照] を選択します。
2. 参照リストに Fortran プロジェクトがある場合は、プロジェクトを選択してから [参照の削除] をクリックします。参照リストにあるすべての Fortran プロジェクトに対してこの操作を行います。[OK] をクリックします。

3. ほかの C/C++ プロジェクトでも上記の手順を実行します。

これにより、プロジェクトの依存関係が更新されます。

1. C/C++ プロジェクトを右クリックして、[プロジェクトの依存関係] を選択します。
2. このプロジェクトと依存関係のあるプロジェクトのチェックボックスをすべてオンにします。
3. [OK] をクリックします。
4. 依存関係のあるほかの C/C++ プロジェクトでも上記の手順を実行します。

以前のバージョンの Visual Studio* とは異なり、Visual Studio* 2010 は依存関係のあるプロジェクトの出力ライブラリーを自動でリンクしません。そのため、親プロジェクトのプロパティー・ページで [Linker (リンカー)] > [Additional Directories (追加のライブラリー・ディレクトリー)] からこれらのライブラリーを明示的に追加する必要があります。必要に応じて、Visual Studio* のマクロである `$(ConfigurationName)` と `$(PlatformName)` を使用してパスを指定することができます。次に例を示します。

```
..\FLIB\$(ConfigurationName)\FLIB.lib
```

`$(ConfigurationName)` は `Release` または `Debug` に置換されます。同様に、`$(PlatformName)` は `Win32` または `x64` に置換されます。

3.9 Fortran 2003 および Fortran 2008 機能の概要

インテル® Fortran コンパイラーは、Fortran 2003 の多くの機能をサポートしています。現在サポートしていない Fortran 2003 機能についても、今後サポートしていく予定です。現在のコンパイラーでは、以下の Fortran 2003 機能がサポートされています。

- Fortran 文字セットが次の 8 ビット ASCII 文字を含むように拡張: ~ \ [ ] ` ^ { } | # @
- 最大長 63 文字までの名前
- 最大 256 行の文
- 角括弧 [ ] を (/ /) の代わりに配列の区切り文字として使用可能
- コンポーネント名とデフォルト初期化を含む構造コンストラクター
- 型と文字列長仕様を含む配列コンストラクター
- 名前付き PARAMETER 定数は複素定数の一部
- 列挙子
- 割り当て可能な派生型のコンポーネント
- 割り当て可能なスカラー変数
- 無指定文字長エンティティー
- PRIVATE コンポーネントの PUBLIC 型と PUBLIC コンポーネントの PRIVATE 型
- ALLOCATE と DEALLOCATE の ERRMSG キーワード
- ALLOCATE の SOURCE= キーワード
- 型拡張子
- CLASS 宣言
- 多相型エンティティー
- 継承と関連付け
- 遅延バインディングと抽象型
- 型バインド・プロシージャー
- TYPE CONTAINS 宣言
- ABSTRACT 属性
- DEFERRED 属性
- NON_OVERRIDABLE 属性
- 型バインド・プロシージャーの GENERIC キーワード
- FINAL サブルーチン
- ASYNCHRONOUS 属性および文
- BIND(C) 属性および文
- PROTECTED 属性および文

- VALUE 属性および文
- VOLATILE 属性および文
- ポインター・オブジェクトの INTENT 属性
- 代入文の左辺と右辺の形状または長さが異なる場合に、左辺の割り当て可能な変数を再割り当て (無指定文字長でない場合、`/assume:realloc_lhs` オプションが必要)
- ポインター代入の境界の仕様と境界の再マップ
- ASSOCIATE 構造
- SELECT TYPE 構造
- すべての I/O 文で、次の数値は任意の種類で指定可能: UNIT=, IOSTAT=
- NAMELIST I/O が内部ファイルで許可
- NAMELIST グループのエンティティーの制限の緩和
- 書式付き入出力で IEEE 無限大と NaN の表現方法が変更
- FLUSH 文
- WAIT 文
- OPEN の ACCESS='STREAM' キーワード
- OPEN およびデータ転送文の ASYNCHRONOUS キーワード
- INQUIRE およびデータ転送文の ID キーワード
- データ転送文の POS キーワード
- INQUIRE の PENDING キーワード
- 次の OPEN 数値は任意の種類で指定可能: RECL=
- 次の READ および WRITE 数値は任意の種類で指定可能: REC=、SIZE=
- 次の INQUIRE 数値は任意の種類で指定可能: NEXTREC=、NUMBER=、RECL=、SIZE=
- 開始する新しい I/O が自身以外の内部ファイルを修正しない内部 I/O の場合、再帰 I/O を利用可能
- IEEE 無限大および非数は Fortran 2003 で指定されるフォーマット出力で表示
- BLANK、DECIMAL、DELIM、ENCODING、IOMSG、PAD、ROUND、SIGN、SIZE I/O キーワード
- DC、DP、RD、RC、RN、RP、RU、RZ 書式編集記述子
- I/O フォーマットで、繰り返し指定子が続く場合、P 編集記述子の後のカンマはオプション
- USE 内のユーザー定義演算子名の変更
- USE の INTRINSIC および NON_INTRINSIC キーワード
- IMPORT 文
- 割り当て可能なダミー引数
- 割り当て可能な関数結果
- PROCEDURE 宣言
- プロシージャー・ポインター
- ABSTRACT INTERFACE
- PASS 属性と NOPASS 属性
- SYSTEM_CLOCK 組込み関数の COUNT_RATE 引数が任意の種類の REAL で指定可能
- STOP 文の実行で IEEE 浮動小数点例外が発生すると警告を表示
- `/assume:noold_maxminloc` が指定された場合、ゼロサイズの配列の MAXLOC または MINLOC でゼロを返す
- 型問い合わせ組込み関数
- COMMAND_ARGUMENT_COUNT 組込み関数
- EXTENDS_TYPE_OF と SAME_TYPE_AS 組込み関数
- GET_COMMAND 組込み関数
- GET_COMMAND_ARGUMENT 組込み関数
- GET_ENVIRONMENT_VARIABLE 組込み関数
- IS_IOSTAT_END 組込み関数
- IS_IOSTAT_EOR 組込み関数
- MAX/MIN/MAXVAL/MINVAL/MAXLOC/MINLOC 組込み関数 (CHARACTER 引数)

- MOVE_ALLOC 組込み関数
- NEW_LINE 組込み関数
- SELECTED_CHAR_KIND 組込み関数
- 次の組込み関数においてオプションで KIND= 引数を指定可能:ACHAR、COUNT、IACHAR、ICHAR、INDEX、LBOUND、LEN、LEN、TRIM、MAXLOC、MINLOC、SCAN、SHAPE、SIZE、UBOUND、VERIFY
- ISO_C_BINDING 組込みモジュール
- IEEE_EXCEPTIONS、IEEE_ARITHMETIC、IEEE_FEATURES 組込みモジュール
- ISO_FORTRAN_ENV 組込みモジュール

このリリースではまだ実装されていないか、動作しない Fortran 2003 機能の一部を次にリストします。

- ユーザー定義の派生型 I/O
- パラメーター化された派生型
- CLASS オブジェクトのデフォルトの初期化
- MODULE PROCEDURE でのキーワード MODULE の省略
- 初期化式での変形組込み関数 (MERGE や SPREAD など) の使用

インテル® Fortran コンパイラーは、Fortran 2008 規格のいくつかの機能もサポートしています。その他の機能は将来のリリースでサポートされる予定です。現在のコンパイラーでは、以下の Fortran 2008 機能がサポートされています。

- 配列の最大次元数が 31 次元に (Fortran 2008 では 15 次元)
- Co-Array
    - CODIMENSION 属性
    - SYNC ALL 文
    - SYNC IMAGES 文
    - SYNC MEMORY 文
    - CRITICAL および END CRITICAL 文
    - LOCK および UNLOCK 文
    - ERROR STOP 文
    - ALLOCATE および DEALLOCATE で Co-Array を指定
    - 組込みプロシージャー: IMAGE_INDEX、LCOBOUND、NUM_IMAGES、THIS_IMAGE、UCOBOUND
    - **注:** ATOMIC_DEFINE および ATOMIC_REF はサポートしていません
- CONTIGUOUS 属性
- ALLOCATE の MOLD キーワード
- DO CONCURRENT
- OPEN の NEWUNIT キーワード
- G0 および G0.d フォーマット編集記述子
- 無制限のフォーマット項目繰り返しカウント指定子
- CONTAINS セクションは空にすることも可能
- 組込みプロシージャー: BESSEL_J0、BESSEL_J1、BESSEL_JN、BESSEL_YN、BGE、BGT、BLE, BLT、DSHIFTL、DSHIFTR、ERF、ERFC、ERFC_SCALED、GAMMA、HYPOT、IALL、IANY、IPARITY、IS_CONTIGUOUS、LEADZ、LOG_GAMMA、MASKL、MASKR、MERGE_BITS、NORM2、PARITY、POPCNT、POPPAR、SHIFTA、SHIFTL、SHIFTR、STORAGE_SIZE、TRAILZ
- 組込みモジュール ISO_FORTRAN_ENV の追加: ATOMIC_INT_KIND、ATOMIC_LOGICAL_KIND、CHARACTER_KINDS、INTEGER_KINDS、INT8、INT16、INT32、INT64、LOCK_TYPE、LOGICAL_KINDS、REAL_KINDS、REAL32、REAL64、REAL128、STAT_LOCKED、STAT_LOCKED_OTHER_IMAGE、STAT_UNLOCKED
- ALLOCATABLE または POINTER 属性を持たない OPTIONAL 仮引数は、対応する実引数に ALLOCATABLE 属性があるのに割り当てられない場合、POINTER 属性があるのに関

連付けが解除されている場合、または NULL 組込み関数への参照の場合、無視されます。
- 仮引数がプロシージャー・ポインターの場合、そのポインターの有効な参照先か、または組込み関数 NULL への参照である実引数に関連付けられます。実引数がポインターではない場合、仮引数に INTENT (IN) 属性が含まれていなければなりません。

4 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。問題の修正の詳細は、http://intel.ly/neQlw2 (英語) を参照してください。

4.1 インテル® MKL 10.3 Update 10 の新機能

- 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) 対応の 32 ビット・プログラムで dznrm2 および dnrm2 のパフォーマンスが向上
- LAPACK: LAPACK バージョン 3.4.0 のサポートを追加
- データ適合: インテル® Xeon® プロセッサー E7-4870/E5-2690 において、以下の領域における SearchCells1D() 関数のパフォーマンスが向上。
  - 補間点の数が 32 を超える任意の非一様および擬一様領域
  - 補間点の数が 32 未満のすべての種類の領域

4.2 インテル® MKL 10.3 Update 9 の新機能

- LAPACK: 非常に小さなサイズ (~10 x 10) で [C/Z]GEEV のパフォーマンスが向上
- FFT: 大幅なパフォーマンス向上のためにインプレースの 1D FFT 実関数をスレッド化
- FFT: 32 ビットのオペレーティング・システムを実行するインテル® Xeon® プロセッサー E5 ファミリーのシステムにおける 2 の累乗の倍精度複素 1D FFT のスケーラビリティー向上のために新しいアルゴリズムを追加
- 乱数ジェネレーター: 新しいインテル® マイクロアーキテクチャー (開発コード名: Ivy Bridge) ベースのプロセッサーで利用可能なハードウェアをサポートする、RdRand 命令に基づく非決定性乱数ジェネレーターをサポート
- ベクトル算術関数: インテル® Core™ プロセッサーにおいて、Erf() 関数および Pow3o2() 関数のパフォーマンスが向上
- データ適合: インテル® Xeon® プロセッサー 5600/7500 番台およびインテル® Core™ i7-2600 プロセッサーにおいて、スプラインベースの評価、微分、積分ルーチンのパフォーマンスが向上
- インテル® MKL Windows* 版の OpenMP* スタティック・ランタイム・ライブラリーを将来のライブラリー・リリースで削除予定

4.3 インテル® MKL 10.3 Update 8 の新機能

- データ適合コンポーネント: ベクトルスプラインの構築、セル探索や二分探索、スプライン補間の評価、微分、積分の 1 次元アルゴリズムをカバーする新しいデータ適合関数のセットを追加。以下のサポートを含む。
  - 1 次スプライン、2 次スプライン、3 次スプライン、ステップワイズ法、ユーザー定義スプライン
  - 最適なパフォーマンスのために構成パラメーターを用いたセル探索
  - ユーザー定義補間 (内挿) および補外 (外挿)
  - ベクトル値関数
  - 列優先および行優先格納形式
- スパース 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) 対応のコア数の多いシステムにおいて、非常に疎な行列の CSR (Compressed Sparse Row) 形式の行列-ベクトル乗算 (?CSRMV) のパフォーマンスが向上
- FFT: インテル® AVX 対応システムにおいて、1D 倍精度 FFT のパフォーマンスが向上

- 統計関数: インテル® Core™ プロセッサーにおいて、分散共分散行列および相関行列の計算 (FAST 法) のパフォーマンスとスケーラビリティーが向上
- スタティック・ライブラリーからカスタム DLL を構築するための Microsoft* Visual Studio* プロジェクト・ツールを追加
- 問題の修正

4.4 インテル® MKL 10.3 Update 7 の新機能

- BLAS: 最近のすべてのインテル® Xeon® プロセッサーにおいて、出力行列が小さく外積が大きい (つまり、入力行列が矩形) の場合に DSYRK/SSYRK のマルチスレッド・パフォーマンスが向上
- BLAS: 最近のすべてのインテル® Xeon プロセッサーにおいて、小さな問題 (<10、beta =1) で ?GEMM のパフォーマンスが向上
- BLAS: 32 ビットのプログラムを実行するインテル® Xeon プロセッサー 5500/5600/7500 番台 において、INCX=1 の小さな問題で DSCAL のパフォーマンスが向上
- BLAS のような拡張: 正方行列のインプレース転置のキャッシュ効率とスレッディングの向上
- PARDISO: PARDISO 用の独立したスレッド化コントロールの追加; mkl_domain_set_num_threads() 関数での MKL_DOMAIN_PARDISO の使用
- ポアソン・ライブラリー: 2D/3D 周期的境界条件のサポートの追加
- ドキュメント・ディレクトリーへのリンク・ライン・アドバイザーの追加
- libtool などのスクリプトツールとともに使用するコマンドライン・リンク・ツールの追加
- 次のコンポーネントの関数の stdcall プロトタイプを含む C ヘッダーファイルの追加: BLAS、スパース BLAS、LAPACK、PARDISO/DSS、RCI 反復ソルバー、ベクトル数学関数、ベクトル・スタティスティカル関数、およびサポート関数
- mkl_domain_set_num_threads() 関数でドメインの指定に使用する定数の名前を変更 (例: MKL_BLAS は MKL_DOMAIN_BLAS に変更); MKL_PARDISO を除き、古い名前も継続して使用可能
- 問題の修正

4.5 インテル® MKL 10.3 Update 6 の新機能

- スパース BLAS: mkl_?csrbsr 変換関数に BSR 形式から CSR 形式への変換時にゼロの要素を検出し削除する新しいオプションを追加
- Windows* での DLL ロード動作の変更: インテル® MKL の DLL を PATH 上の個別のディレクトリーに配置することはできなくなりました。すべての DLL を実行ファイルと同じディレクトリーに配置するか、PATH 環境変数で指定されている別のディレクトリーに配置する必要があります。
- 問題の修正

4.6 インテル® MKL 10.3 Update 5 の新機能

- BLAS: パフォーマンスの向上: {S,C,Z}TRSM (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) 対応プロセッサー); {S,D}GEM2VU (インテル® AVX 対応プロセッサー、インテル® Core™ i7 プロセッサー、インテル® Xeon® プロセッサー 5500 番台)
- BLAS: スケーリングの向上: ?TRMV (すべてのアーキテクチャーの大規模行列); DGEMM (インテル® Xeon® プロセッサー 5400 番台でスレッド数が奇数の場合)
- LAPACK: LAPACK 3.3.1 の拡張とそれぞれの LAPACKE インターフェイスに対応
- LAPACK: 一般固有値問題に使用される ?SYGST と ?HEGST のパフォーマンスが向上
- LAPACK: LU 分解された行列の逆行列のパフォーマンスが向上 (?GETRI)
- PARDISO: 転置と共役転置の解算出 (ATx=b と AHx=b) の追加; 圧縮されたスパース列 (CSC) 形式のサポート

- PARDISO: MKL_PARDISO_OOC_MAX_SWAP_SIZE 環境変数とインコア PARDISO を使用して、必要なメモリー量が利用可能なメモリー量をやや上回る場合に、アウトオブコア PARDISO のパフォーマンスが向上
- 最適化ソルバー: RCI Trust-Region ソルバーに Inf と NaN のチェックを追加
- FFT: インテル® SSE3 以降に対応したインテル® プロセッサーにおいて、サポートされているすべての精度で 2x2x2 から 10x10x10 の小さな立方体に対する 3D FFT のパフォーマンスが向上
- FFT サンプル: インテル® MKL の DFTI と FFTW の一般的な使用例を示すサンプルコードを変更
- VSL: インテル® Core™ i7-2600 プロセッサーを搭載した 64 ビットのオペレーティング・システムのマシンにおいて、単精度の MT19937 および MT2203 基本乱数ジェネレーターのパフォーマンスが向上
- VSL: インテル® Core™ i7-2600 プロセッサーおよびインテル® Xeon® プロセッサー 5400 番台で SOBOL 準乱数ジェネレーターの整数バージョンのパフォーマンスが向上。

4.7 インテル® MKL 10.3 Update 4 の新機能

- BLAS: インテル® Xeon® プロセッサー 5400 番台以降において DTRMM のパフォーマンスが向上
- BLAS: すべての 64 ビット対応プロセッサー、特にインテル® アドバンスト・ベクトル・エクステンション (インテル® AVX) 対応プロセッサーにおいて DTRSM のパフォーマンスが向上
- LAPACK: LAPACK 3.3.1 リリースでの問題の修正に対応
- OOC PARDISO: アウトオブコア処理に必要なメモリー量の推定が向上
- FFT: スレッディングの改善による 1D 実数 FFT スケーリングの向上
- FFT: 新しいシングル・ダイナミック・ライブラリー・リンク・モデルを使用するように C および Fortran の FFT サンプルを更新
- VML: インテル® Xeon® プロセッサー 5600/7500 番台およびインテル® Core™ i7-2600 プロセッサーにおいて、すべての精度で、Hypot 実関数と Abs 複素関数の単精度のパフォーマンス拡張バージョン、および Arg、Div、Mul、MulByConj 複素関数のパフォーマンスが向上
- サービス関数: インテル® MKL のサービス関数の追加と拡張 (詳細は、http://intel.ly/pkUQXI (英語) のオンライン・リリースノートを参照してください)
- 問題の修正

4.8 インテル® MKL 10.3 Update 3 の新機能

- BLAS: インテル® Xeon® プロセッサー 5400 番台を搭載した 32 ビットの Windows* システムにおいて DSYRK、DTRSM、DGEMM のマルチスレッド・パフォーマンスが向上
- LAPACK: 対称/エルミート行列関数および補助関数における連立線形方程式ソルバーの向上、CS (余弦/正弦) 分解を含む Netlib LAPACK 3.3 の実装
- PARDISO: 0 ベースの順列ベクトルの入力をサポート
- PARDISO: pardisoinit() ルーチンのドキュメント化
- PARDISO: 複数の右辺 (RHS) を含む PARDISO のシリアル・パフォーマンスが向上
- PARDISO: 小さな行列のパフォーマンスを向上させる解のステップの並列化の独立制御。詳細は、iparm(25) の説明を参照してください。
- PARDISO: 後方代入の減少による右辺全体の部分解計算。詳細は、iparm(31) の説明を参照してください。
- FFT: 最大 3 ～ 7 次元の実数 FFT 変換の実装
- FFT: 2 つの実数配列として表される分割複素数データを使用した多次元複素数変換の並列化
- クラスター FFT: FORTRAN 90 インターフェイスの拡張による実数-複素数変換への対応、および新しいサンプルの追加

- VML: 新しい Pack/Unpack 複素関数と Gamma/LGamma 実関数の追加
- VML: インテル® Xeon® プロセッサー 5600 番台およびインテル® アドバンスト・ベクトル・エクステンション (インテル® AVX) 対応プロセッサーで、すべての関数におけるショートベクトル (< 100) の演算、すべての関数におけるアライメントされていない入力ベクトルの演算、sPow2o3 関数、拡張パフォーマンス (EP) バージョンの Add および Sub 複素関数のパフォーマンスが向上
- VSL: 乱数ジェネレーター (RNG) ストリームからメモリーへの保存、またはメモリーからの復元を行うための関数の追加
- VSL: 新しい UniformBits32 関数および UniformBits64 関数の追加
- VSL: MT2203 BRNG でサポートされる一意のストリーム数を 1024 から 6024 に拡張
- 問題の修正

注: インテル® MKL に含まれる GMP* 数学関数は、将来のリリースでは削除されます。

4.9 インテル® MKL 10.3 Update 2 の新機能

- BLAS: インテル® Xeon® プロセッサー 5600 番台において転置関数のパフォーマンスが向上
- BLAS: 転置ルーチンのサンプルの追加
- FFT: 必要な精度の関数のみをリンクすることでアプリケーションのフットプリントを小さくする方法を示した Fortran サンプルの追加
- FFT: CCE ストレージを使用するインプレース実数変換にストライドの一貫性チェックを追加
- FFT: 多次元変換のスレッド化の追加
- VSL: クアッドコア インテル® Xeon® プロセッサー 5500 番台において単精度/倍精度の多変量ガウス分布乱数ジェネレーターのパフォーマンスが向上
- VML: インテル® Xeon® プロセッサー 5500 番台において Add、Mul、Sub 関数のインプレース操作のパフォーマンスが向上
- 問題の修正

4.10 インテル® MKL 10.3 Update 1 の新機能

- PARDISO/DSS: F90 オーバーロード API の追加 (詳細は、インテル® MKL リファレンス・マニュアルを参照してください)
- PARDISO: 統計情報をより見やすく改良
- スパース BLAS: 最新のインテル® プロセッサーにおいて ?BSRMM 関数のパフォーマンスが向上
- FFT: 負のストライドのサポート
- FFT サンプル: DFTI と FFTW3 の両方のインターフェイスを使用した分割複素 FFT の C および Fortran サンプルの追加
- VML: SSE2 および SSE3 対応システムにおいて、インプレースの Add/Sub/Mul/Sqr 実関数のパフォーマンスが向上
- ポアソン・ライブラリー: ポアソン・ライブラリー関数のデフォルトの動作をシリアルから並列に変更
- 問題の修正

4.11 インテル® MKL 10.3 の新機能

- 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
- 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
- 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
- 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
- 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
- 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
- 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
- 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

- 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
- 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
- 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
- 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

4.12 既知の問題

本リリースにおける既知の制限事項の詳細なリストは、http://intel.ly/ptEfAP (英語) を参照してください。

4.13 注意事項

インテル® MKL の将来のバージョンでは以下の変更が予定されています。「テクニカルサポート」を参照してください。

- ファイル名に solver を含むライブラリーの内容をコア・ライブラリーに移動する予定です。これらの solver ライブラリーはその後削除される予定です。
- インテル® MKL の GMP (GNU* Multiple Precision) 関数インターフェイスは、将来のリリースでは削除される予定です。
- タイミング関数 mkl_set_cpu_frequency() は古い関数です。代わりに、『インテル® MKL リファレンス・マニュアル』で説明されている mkl_get_max_cpu_frequency()、mkl_get_clocks_frequency()、mkl_get_cpu_frequency() を使用してください。
- PARDISO ドメインを指定するために定義されている MKL_PARDISO 定数は、mkl_domain_set_num_threads() 関数で使用できなくなりました。代わりに、MKL_DOMAIN_PARDISO を使用してください。
- 畳み込みルーチンと相関ルーチンは、将来のリリースでは 10.2 Update 3 との下位互換性はなくなります。

4.14 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。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。

5 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。

5.1 新機能

- [Data Sharing Events (データ共有イベント)] ウィンドウで、ローカル・ステータス・バーにデータ共有検出の状態が表示されるようになりました。

5.2 既知の問題

- Co-Array の要素を表示できません。
- Fortran の多次元配列が適切に表示されません。また、フィルター式でも受け付けません。
- Fortran の複素数型が適切に表示されません。
- カスタムの配列境界による Fortran 配列にフィルターを設定できません。
- 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] > [構成プロパティ] > [Fortran] > [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* 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。
- 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。
- インテル® Parallel Debugger Extension がインストールされている場合、Experimental Hive モードでアプリケーションをデバッグすると、Microsoft* Visual Studio* 2010 デバッガーがクラッシュすることがあります。現在この問題の解決方法はまだありません。クラッシュした場合は、インテル® Parallel Debugger Extension をアンインストールする必要があります。[コントロール パネル] の [プログラムの追加と削除] または [プログラムと機能] から、インテル(R) Visual Fortran Composer XE 2011 <version> Windows* 版を選択して、[変更と削除] をクリックします。[変更] ダイアログで、インテル(R) Parallel Debugger Extension の選択を解除してください。

5.3 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。

6 著作権と商標について

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、インテルの Web サイトを参照してください。

http://www.intel.com/design/literature.htm

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。

Intel、インテル、Intel ロゴ、Intel Core、Itanium、Pentium、Xeon は、アメリカ合衆国および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。