

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
2011 年 3 月 1 日

目次

1	概要	3
1.1	変更履歴	3
1.2	製品の内容	3
1.3	動作環境	4
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	5
1.4	ドキュメント	5
1.5	サンプル	6
1.6	日本語サポート	6
1.7	テクニカルサポート	7
2	インストール	7
2.1	インストール前の準備	7
2.1.1	64 ビット・アプリケーション用の Visual Studio* の設定	7
2.1.2	Microsoft* Windows Vista* および Microsoft* Windows* 7 でのインストール	7
2.2	インストール	8
2.2.1	インテルのアクティベーション・ツールを使用した製品のアクティベーション ...	8
2.2.2	ライセンスサーバーの使用	8
2.2.3	IMSL Fortran 数値計算ライブラリーのインストール	8
2.2.4	Microsoft* Visual Studio* 2005 の管理者権限に関するメッセージダイアログ	8
2.3	製品の変更、更新、削除	9
2.4	サイレント・インストール/アンインストール	10
2.5	インストール先フォルダー	10
2.6	インストールの既知の問題	11
2.6.1	Microsoft* Visual Studio* 2010 用ドキュメントをインストールするための追加ステップ	11
2.6.2	複数の Visual Studio* のバージョンを使用している場合のドキュメントに関する問題	11

3	インテル® Visual Fortran コンパイラー	11
3.1	互換性.....	12
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
3.2	新規および変更されたコンパイラー機能	12
3.2.1	Fortran 2003 の機能.....	12
3.2.2	Fortran 2008 の機能.....	12
3.2.3	Co-Array	13
3.2.4	その他の変更	14
3.3	新規および変更されたコンパイラー・オプション	14
3.3.1	/Qsox オプションの追加キーワード、デフォルトの変更 (12.0.3).....	15
3.4	その他の変更.....	16
3.4.1	ビルド環境コマンドスクリプトの変更.....	16
3.4.2	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.....	16
3.4.3	OpenMP* レガシー・ライブラリーの削除	17
3.4.4	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.....	17
3.4.5	IMSL の再配布に関するライセンスの変更	17
3.4.6	RANF 移植関数の組み込み関数への変更	18
3.5	既知の問題	18
3.5.1	日本語ファイル名に関するコマンドライン診断表示の問題.....	18
3.6	Microsoft* Visual Studio* 2010 に関する注意事項	18
3.6.1	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.....	18
3.6.2	プロジェクトの依存関係の調整	20
3.7	Fortran 2003 および Fortran 2008 機能の概要	20
4	インテル® マス・カーネル・ライブラリー	23
4.1	インテル® MKL 10.3 Update 3 の新機能	23
4.2	インテル® MKL 10.3 Update 2 の新機能	24
4.3	インテル® MKL 10.3 Update 1 の新機能	24
4.4	インテル® MKL 10.3 の新機能.....	24
4.5	既知の問題	26
4.6	注意事項	26
4.7	権利の帰属	26
5	インテル® Parallel Debugger Extension	27
5.1	新機能.....	27
5.2	既知の問題	27

5.3	ドキュメント	28
6	著作権と商標について	29

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 3 (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 (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 (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® Visual 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 12.0.3。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 3
- Microsoft* 開発環境への統合
- Microsoft* Visual Studio* 用インテル® Parallel Debugger Extension 12.0.3
- Microsoft* Visual Studio* 2008 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 Windows* 版 IMSL* 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://software.intel.com/en-us/articles/intel-architecture-platform-terminology> (英語) を参照してください。

- インテル® ストリーミング SIMD 拡張命令 2 (インテル® SSE2) 対応の IA-32 またはインテル® 64 アーキテクチャー・プロセッサをベースとするコンピューター (インテル® Pentium® 4 プロセッサ以降、または互換性のあるインテル以外のプロセッサ)
 - 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・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、Microsoft* Windows Vista*、Microsoft* Windows* 7、Microsoft* Windows Server* 2003、Microsoft* Windows Server* 2008、Microsoft* Windows HPC Server* 2008 (エンベデッド・エディションはサポートされていません)
 - Microsoft* Windows Server* 2008 または Windows HPC Server 2008 では Microsoft* Visual Studio* 2008 SP1 あるいは 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* 2008 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]
- 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++* 2008 Express Edition [3]
 - 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
- 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* 2008 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 Studio* Shell をインストールし、また Microsoft* Visual C++* 2008 Express Edition (Microsoft* C++ コンパイラへの別アクセス) も使用する場合は、インテル® Visual Fortran コンパイラと Visual Studio* Shell をインストールする前に Visual C++* 2008 Express Edition をアンインストールしてください。Fortran のインストールが完了したら、必要に応じて Visual C++* 2008 Express Edition をインストールします。Fortran と C++ コンパイラ環境は個別で、混合はされません。
4. インテル® Visual Fortran コンパイラは、デフォルトで、インテル® SSE2 命令対応のプロセッサが必要な 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・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 11.1 ではまだサポートされています。

1.4 ドキュメント

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

最適化に関する注意事項

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

インテル® コンパイラー、関連ライブラリーおよび関連開発ツールは、互換マイクロプロセッサer向けには、インテル製マイクロプロセッサer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)、インテル® ストリーミング SIMD 拡張命令 3 補足命令 (インテル® SSSE3) 命令セットに関連する最適化およびその他の最適化が含まれます。インテルでは、インテル製ではないマイクロプロセッサer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er固有の最適化は、インテル製マイクロプロセッサerでの使用を目的としています。

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

改訂 #20110228

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。

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

1.7 テクニカルサポート

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

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

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

2 インストール

2.1 インストール前の準備

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

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

Visual Studio* 2005/2008 Standard Edition または Visual Studio* 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。

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

Microsoft* Visual Studio* 2005 ユーザーは、Visual Studio* 2005 Service Pack 1 (VS 2005 SP1) と Visual Studio* 2005 Service Pack 1 Update for Windows Vista* (VS 2005 SP1 ページからリンクが提供) をインストールしてください。これらのアップデートをインストールした後に、管理者権限で Visual Studio* が実行できることを確認してください。実行できない場合、インテル® コンパイラを使用できません。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。

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

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

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

2.2.2 ライセンスサーバーの使用

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

2.2.3 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 同梱版のライセンスをお持ちの場合、コンパイラとは別に 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をインストールしてください。

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

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



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

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

2.3 製品の変更、更新、削除

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

コンパイラーのインストールの一部として Microsoft* Visual Studio* 2008 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 2008 Shell ENU
- Microsoft Tools and Libraries for Intel(R) Visual Fortran

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。バージョン 11.0 よりも古いコンパイラーは Visual Studio* から選択できません。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://software.intel.com/en-us/articles/fortran-windows-silent-installation-guide/> (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。

- C:\Program Files\Intel\ComposerXE-2011
 - bin
 - ia32
 - ia32_intel64
 - intel64
 - compiler
 - include
 - ia32
 - intel64
 - lib
 - ia32
 - intel64
 - debugger
 - Documentation
 - help
 - mkl
 - benchmarks
 - bin
 - examples
 - include
 - interfaces
 - lib
 - tests
 - tools
 - redistrib
 - 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 フォルダ一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、フォルダ一名は Program Files (X86) またはそれに相当する名前です。

デフォルトでは、アップデートによって既存のディレクトリーの内容が置換されます。最初のアップデートをインストールするときに、以前のインストールとは別に新しいアップデートをインストールして、システムに両方のファイルを残すオプションを選択できます。両方を残す

オプションを選択した場合、古いアップデートのトップレベルのフォルダー名は ComposerXE-2011.nnn (nnn はアップデート番号) に変更されます。

2.6 インストールの既知の問題

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

Microsoft* Visual Studio* 2010 がインストールされているシステムにインテル® 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/ja-jp/library/dd264831.aspx> を参照するか、「ヘルプ ライブラリ マネージャー」で検索してください。

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

システムに Microsoft* Visual Studio* 2005 と 2008 の両方をインストールしていてインテル® 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。

- 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。
- REAL(16)、REAL*16、COMPLEX(16)、または 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

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 (組み込み関数を含む) 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

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

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
- 組み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
- GO および GO.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
- 組み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
- 組み込みモジュール 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

3.2.3 Co-Array

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。インテル® クラスタ・ツールキットをインストールすると、分散メモリーでの実行に必要なインテル® MPI ランタイム・ライブラリーがインストールされます。別の MPI 実装または OpenMP* を使用する Co-Array アプリケーションはサポートしていません。

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 の共有または分散メモリー処理

ドキュメントでは、`/Qcoarray` オプションは次のように説明されています。

引数なしで `/Qcoarray` (Windows*) または `-coarray` (Linux*) を使用することは、インテル® クラスタ・ツールキットのライセンスがインストールされている場合にマルチノード (分散メモリー) で実行するのと、インテル® クラスタ・ツールキットのライセンスがインストールされていない場合にシングルノード (共有メモリー) で実行するのと同じです。

このドキュメントが記述された後に Co-Array の実装が変更されました。新しい実装では、`/Qcoarray` が `memory` 引数なしで指定された場合、インテル® Cluster Toolkit のライセンスの

有無にかかわらず、共有メモリーが使用されます。分散メモリーを使用するには、`/Qcoarray:distributed` を指定します。この場合、インテル® Cluster Toolkit のライセンスが必要になります。

3.2.3.2 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。

- 分散メモリー環境における Co-Array
- 文字データ型 Co-Array
- ALLOCATABLE または POINTER 属性を含む最終コンポーネントを持つ派生型の Co-Array
- 別のイメージを参照している Co-Array の配列スライス出力 (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
- 別のイメージでの LOCK/UNLOCK の使用
- LOCK、UNLOCK、SYNC IMAGES、SYNC MEMORY、SYNC ALL での STAT= または ERRMSG= 引数の設定

3.2.4 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。

- `/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。

3.3.1 /Qsox オプションの追加キーワード、デフォルトの変更 (12.0.3)

オブジェクト・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 オプションで、インライン展開された関数のリストを含め、プロシージャーのプロファイル情報を含めないように指定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を得るためには、/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 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。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。

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

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

3.4.2.1 スタティック・セキュリティ解析の動作変更

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witch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witch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witch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。

3.4.3 OpenMP* レガシー・ライブラリーの削除

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

3.4.4 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。

3.4.5 IMSL の再配布に関するライセンスの変更

インテル® Visual Fortran Composer XE 2011 Windows* 版 IMSL* 同梱製品のリリースより、IMSL を使用してビルドしたアプリケーションを配布する場合は、IMSL のランタイムライセンスを購入する必要があります。当該アプリケーションを社内またはエンティティーに配布する場合は、

このライセンスをインテルより購入できます。当該アプリケーションをサードパーティーに配布する場合は、Visual Numerics, Inc. に直接問い合わせる必要があります。IMSL ランタイムライセンスの購入に関する詳細は、次のリンクを参照してください。

<http://software.intel.com/en-us/articles/buy-or-renew/>

IMSL ライブラリーには、Visual Numerics, Inc. のエンド・ユーザー・ソフトウェア使用許諾契約書の諸条件が適用されます。IMSL ライブラリーは、コンパイラーの再配布可能ライブラリー・パッケージには含まれていません。IMSL を使用するアプリケーションを開発する場合は、アプリケーションに必要な再配布可能ファイルを特定し、これらのファイルをアプリケーションのインストール・パッケージとは別に含める必要があります。

3.4.6 RANF 移植関数の組み込み関数への変更

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み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、標準の組み込み関数 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。

3.5 既知の問題

3.5.1 日本語ファイル名に関するコマンドライン診断表示の問題

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

3.6 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。

3.6.1 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せん。

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・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・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・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\\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\\Local Settings\AppData\Local\Microsoft\MSBuild\v4.0
```

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。

3.6.2 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・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。

3.7 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
- 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
- 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
- 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・ポインター
- ABSTRACT INTERFACE
- PASS 属性と NOPASS 属性
- SYSTEM_CLOCK 組み込み関数の 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
- 型問い合わせ組み込み関数
- COMMAND_ARGUMENT_COUNT 組み込み関数
- EXTENDS_TYPE_OF と SAME_TYPE_AS 組み込み関数
- GET_COMMAND 組み込み関数
- GET_COMMAND_ARGUMENT 組み込み関数
- GET_ENVIRONMENT_VARIABLE 組み込み関数
- IS_IOSTAT_END 組み込み関数
- IS_IOSTAT_EOR 組み込み関数
- MAX/MIN/MAXVAL/MINVAL/MAXLOC/MINLOC 組み込み関数 (CHARACTER 引数)
- MOVE_ALLOC 組み込み関数
- NEW_LINE 組み込み関数
- SELECTED_CHAR_KIND 組み込み関数
- 次の組み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 組み込みモジュール
- IEEE_EXCEPTIONS、IEEE_ARITHMETIC、IEEE_FEATURES 組み込みモジュール
- ISO_FORTRAN_ENV 組み込みモジュール

サポートされていない Fortran 2003 機能には次の項目が含まれます。

- 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
- ALLOCATE での多相の SOURCE= の指定

インテル® Fortran コンパイラーは、Fortran 2008 標準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
- 組み込みプロシージャ: 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
- GO および GO.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
- 組み込みプロシージャ: 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
- 組み込みモジュール 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

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://software.intel.com/en-us/articles/intel-mkl-103-bug-fixes/> (英語) を参照してください。

4.1 インテル® MKL 10.3 Update 3 の新機能

- BLAS: インテル® Xeon® プロセッサ 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® アドバンスド・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。

4.2 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

4.3 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

4.4 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(*GEQRF)、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と部分解ベクトルを出力する部分解算の追加
 - 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

4.5 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://software.intel.com/en-us/articles/intel-mkl-kb/all> (英語) を参照してください。

4.6 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。

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

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

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

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。

```
ala01db0 Intel Parallel Debugger Extension Exception 0
ala01db1 Intel Parallel Debugger Extension Exception 1
ala01db2 Intel Parallel Debugger Extension Exception 2
ala01db3 Intel Parallel Debugger Extension Exception 3
ala01db4 Intel Parallel Debugger Extension Exception 4
ala01db5 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。

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

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てしないでください。

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ます。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・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・ナンバーは同一プロセッサ・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・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、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。

© 2011 Intel Corporation. 無断での引用、転載を禁じます。