

PGI (Accelerator) Visual Fortran[®] 2018

For Microsoft[®] Visual Studio

Windows[®]版 (Release 2018) - 入門ガイド -

2018年2月版 (Rev. 18.1-A)

株式会社 ソフテック HPC ソリューション部

(http://www.softek.co.jp/SPG/)



1	は	じめに	. 1
	1.1 1.2 1.3 1.4	本文書の概要. WINDOWS 上での PVF ソフトウェアの実装. PVF コンパイラの利用方法 . PVF コンパイラのコマンド・オプションについて.	. 1 . 1 . 1 . 1
2	P۷	F コンパイラの起動 (Microsoft [®] Visual Studio 統合)	. 3
	 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 	VISUAL STUDIO 2015 の初回起動. プロジェクトの作成方法. PVFプロジェクト・テンプレート. 新規にプロジェクトを作成し、新規にプログラム開発するための手続き. 既存のソースファイルを PVF プロジェクトに移行するための手続き. プログラムのコンパイルと実行(デバッグモード) プログラムのコンパイルと実行(最適化オプションの適用) プログラムの実行(入力データファイルのリダイレクト) MPI プログラムのビルド	. 4 . 4 . 6 . 7 10 14 18 25 27
3	GP	U用 OpenACC と CUDA Fortran を使用する	30
	3.1 3.2	OPENACC ディレクティブの利用 PGI CUDA FORTRAN のコンパイル	30 34
4	PV	F コンパイラの起動 (コマンド・ライン)	36
	4.1 4.2 4.3	PVFコマンドプロンプトの起動 PVFコンパイラ・コマンドの使用 WINDOWS®上で使用する際の留意点	36 37 39
5	その	の他	40
	5.1 5.2	実行モジュールの再配布 PVF ドキュメント	40 40

本資料の全ての情報は、現状のまま提供されます。株式会社ソフテックは、本資料に記述あるいは表現されている 情報及びその中に非明示的に記載されていると解釈されうる情報に対して一切の保証をいたしません。また、本資 料に含まれる情報の誤りや、それによって生じるいかなるトラブルに対しても一切の責任と補償義務を負いません。 また、本資料に掲載されている内容は、予告なく変更されることがあります。 本資料で使用されている社名、製品名などは、一般に各社の商標または登録商標です。

株式会社ソフテック 〒 154-0004 東京都世田谷区太子堂 1-12-39 http://www.softek.co.jp

Copyright © 2018 SofTek Systems, Inc. All rights reserved.

1 はじめに

1.1 本文書の概要

本文書は、Microsoft[®] Visual Studio 2015 による統合開発環境(IDE)上で使用する PGI[®] Visual Fortran コンパイラ(以下、「PVF」と言う。)の一般的な使用方法を簡 単に纏めた入門ガイドです。Visual Studio の細かな操作方法に関しては、Microsoft[®] 社のドキュメント等を参考にしてください。

【ご注意】

Microsoft[®] Visual Studio 2008~2012 にプラグイン可能な PGI Visual Fortran は、 PVF 15.10(2015 年)以前のバージョンのソフトウェアとなります。PVF 2016 (2016 年 3 月リリース)以降は、Visual Studio 2008~2012 上で使用できませんのでご了 承下さい。

1.2 Windows 上での PVF ソフトウェアの実装

Microsoft[®] Visual Studio 2015 が実装されているシステム上で PVF ソフトウェア のインストールを行うと、以下のディレクトリ・パス上にソフトウェアが実装され ます。PVF のコンポーネントのデフォルトのインストール・パスは、以下の形態と なります。PVF ソフトウェアを構成するコンポーネントは以下の二つに大別されま す。

- Visual Studio に統合するための PVF モジュール (PVF IDE 部と言う)
- PGI コンパイラ本体のコンポーネント

なお、PGI 2018 以降、Windows 32 ビット用ソフトウェアは提供されません。

【Win64 システム上】

C:¥Program Files (x86)¥Microsoft Visual Studio 14¥PGI Visual Fortran C:¥Program Files¥PGI (64 ビット PGI コンパイラ本体)

1.3 PVF コンパイラの利用方法

PVF コンパイラを使用する形態は、以下の二つの方法があります。PVF は、統合 開発環境(IDE)上でコンパイラの利用できるだけではなく、コマンド・ライン上でも 利用可能です。

	「コンハイノの利用形態					
利用方法	内容					
Visual Studio の統合開発環境	Visual Studio を起動して、IDE 上の GUI ベース ⁻					
(IDE)上での使用	操作する。					
コマンド・ライン上での使用	PVF Command Prompt(64bit)のウィンドウを開					

PVF コンパイ	'ラ	の利用形態
----------	----	-------

1.4 PVF コンパイラのコマンド・オプションについて

PGI コンパイラには、本 PVF コンパイラ製品だけではなく、Linux/Windows/Apple

き、コマンドベースでコンパイラを操作する。

macOS 等の OS 下においてコマンド・ライン上で操作する「PGI Compilers 製品」 があります。これらの製品の種別を問わず、PGI コンパイラで使用するコンパイラ・ オプション (スイッチ) 名とその使用方法は、一部、OS に依存したものを除き、 同じものとお考えください。PGI コンパイラのオプションの説明に関しては、弊社 ホームページ上の以下のページをご参照ください。

http://www.softek.co.jp/SPG/Pgi/comp-tips.html

例えば、以下の pgfortran コマンドの例は、全ての PGI 製品のコマンド・ライン 上で、全く同じ形で使用できます。PGI コマンド列は、Linux 流のコマンド列コン ベンション(慣用的な使用法)を踏襲しているため、Windows 上においても、コマ ンド・オプションは、「-」で始まる形態となります。(一般的な Microsoft Windows 上でのコマンド・オプションは、その始めに「¥」(バックスラッシュ)を付けて記 述するのが一般的ですが、PVF では、「-」を前に付けて指定します)

\$ pgfortran <u>-fastsse</u> -Minfo test.f90

(「-」で始まるものは、コンパイラ・オプションを意味します)

PGI コンパイラのコンパイルオプションの詳細に関しては、以下の URL をご覧ください。

http://www.softek.co.jp/SPG/Pgi/comp-tips.html

PVFの Visual Studio (IDE)上での使用においても、同じ形態でコンパイラ・オプションがセットされております。また、以下の図のように任意のコンパイラ・オプションを IDE 上の「プロジェクトのプロパティ」でセットすることが可能です。

【テクニック】

以下の画面は、コンパイラのオプション等を設定するプロパティの画面です。 Fortran あるいは Linker の中の個々のプロパティ・ウイザードでオプションを設定 するのが一般的ですが、<u>これが煩わしい場合</u>は、以下のように、「Command Line」 で、必要とする PGI のコマンド・オプションを<u>直に指定することでも代替</u>できます。 この場合は、その内容をプロパティ・ウイザードで設定する必要はありません。な お、Linker の「Command Line」にも同じオプションの指定が必要な場合もありま す。指定するコマンド・オプションは、PGI コンパイラの一般的なオプションの全 てが指定することが出来ます。もちろん、ウイザードで指定したオプションが重複 して指定したとしても問題はありません。

特に、<u>-Minfo オプション</u>は、コンパイル時の最適化メッセージ、並列化メッセージ、OpenACC メッセージ等を「出力ダイアログ」に表示するためのものです。このメッセージによって、最適化されたソース行が理解できるため、常にコマンドラインオプションとして指定しておくことをお勧めします。





出力	→ □ ×
出力元の表示(<u>S</u>): ビルド	- 🖆 🖆 🞽 🔛
 Generated vector sse code for the loop Generated a prefetch instruction for the loop zfft3d0: 85, Loop not vectorized/parallelized: too deeply nested 86, Loop not vectorized/parallelized: too deeply nested 90, Loop unrolled 18 times Generated 4 prefetches in scalar loop 	
95, Loop not vectorized/parallelized: contains call 100, Loop unrolled 16 times	-Minfo によるコンパイルメッセージ
Generated 4 prefetches in scalar loop 107, Loop not vectorized/parallelized: too deeply nested 108, Loop not vectorized/parallelized: too deeply nested 112, Loop unrolled 18 times Generated 4 prefetches in scalar loop 117, Loop not vectorized/parallelized: contains call 122, Loop unrolled 18 times Generated 4 prefetches in scalar loop 127, Loop not vectorized/parallelized: contains call	
4	• • •

© SofTek Systems, Inc

2 PVF コンパイラの起動(Microsoft[®] Visual Studio 統合)

2.1 Visual Studio 2015 の初回起動

PVF コンパイラを Visual Studio 統合環境上で使用する際には、「Visual Studio 2015」を起動する必要があります。以下の Visual Studio の画面が現れます。(Visual Studio の <u>Shell integrated mode</u> を使用する際は、以下の画面は現れません)

Visual Studio Professional Edition を初めて起動した際、以下のような画面が表示 されます。Visual Studioの開発作業環境のデフォルトを指定するものですが、これ は、「全般的な開発設定」を選んで Visual Studioの開始を行ってください。

既定の環境設定の選択	X							
Visual Studio [®] 2010 Profession	nal							
アプリケーションを最初に使用する前に、Visual Basic または Visual C# など最も従事する開発作 業の種類を指定してください。この情報に基づいて、あらかじめ定義された設定のコレクションが ユーザーの開発作業に合わせた環境に適用されます。								
いつでも設定の別のコレクションを使用する。 ンポートとエクスポート]を選択し、[すべての	ように変更できます。 [ツール] メニューの [設定のイ の設定をリセット] を選択します。							
▽ 以前のパージョンから有効な設定を移行し 既定の環境設定を選択してください(C):	、以下で選択した既定の設定と合わせて適用する(<u>M</u>)							
以前の設定	説明:							
Visual Basic 開発設定	アプリケーションの以前のバージョンに近い開発環							
VISUAI C# 開発設定 Visual C++ 開登設定	境を提供します。複数のブログラミング言語で開発							
Visual F# 開発設定	9 る場合は、この設たコレクションを進入してくた さい。							
Web 開発								
Web 開発 (コードのみ)								
ノロシェクト管理の設定 全般的な関発設定								
Visual Studi	io の開始(<u>S)</u> Visual Studio の終了(<u>X</u>)							

2.2 プロジェクトの作成方法

Windows のアプリケーション・メニューから Visual Studio を開始して、新しい 「プロジェクト」を作成します。Visual Studio を起動後、「ファイル」->「プロジェ クト」を選択します。

Visual Studio 上で新規にプログラムを開発・作成する場合も、既存のプログラム を Visual Studio のプロジェクトの中に移行する場合も、この「プロジェクト」作成 メニューを使用します。

×	Microsoft Visual Studio				クイック起動 (C	trl+Q) p = 🗆 ×
77	マイル(F) 編集(E) 表示(V) デ, (ッグ(D)	チーム(M) SQL(())	<u>₩ール(T) 〒7.5(S) 分析(N)</u> ウィンド	ウ(W) ヘルプ(Ή)
	新規作成(N)	• •	18	プロジェクト(P)	Ctrl+Shift+N	1
	閒<(0)				Shift+Alt+N	
	閉じる(C)		E	チーム プロジェクト(T)		
×	ソリューションを閉じる(T)		õ	ファイル(F)	Ctrl+N	
	遥択されたファイルを上書き保存(S)	Ctrl+S		既存のコードからプロジェクトを作成(E)		
	選択したファイルに名前を付けて保存(A)		 _			
6	すべてを保存(L)	Ctrl+Shift+S				
	テンプレートのエクスポート(E)					
	ソース管理(R)	Þ				
₽	ページ設定(U)					
	印刷(P)	Ctrl+P				
	最近使ったプロジェクトとソリューション(J)	Þ				
×	終了(X)	Alt+F4				
	出力 出力元の表示(S):		•	1111 1111 1111 1111 1111 1111 1111 11	• # X	
34 / 22	エラー一覧 出力					コード分析 ソリューション エ…

プロジェクトの新規作成を選択しますと、以下の画面が現れます。画面左側に示 される「PGI Visual Fortran」プロジェクトをクリック選択しますと、右側に PVF 関連の新規「テンプレート」が表示されます。「PGI Visual Fortran」プロジェクト は、64 ビットモード(x64)が表示されます。64 ビットアプリケーション作成用の x64 テンプレートを使用します。以下の画面は、64 ビット Windows x64 上での表示例 を示したものです。(32 ビット Windows の場合は、32 ビットモード(Win32) モ ードのみ表示されます) PVF 2018 以降は、32 ビットモード機能は提供しません。



上図で①の部分は、Visual Studio 上の「プロジェクト名」を指定するものです。 ②は、ソリューション名を指定します。初めてソリューションを作成する時は、「ソ リューションのディレクトリを作成」にチェックを入れます。「場所」で示される部 分は、このプロジェクト関連で作成されるファイルを保存する場所を指定します。 デフォルトは、Windows上の「マイドキュメント」配下に作成されます。この場所 を変えたい場合、あるいはプロジェクト名を変更したい場合は、そのパス名、フォ ルダ名を指定してください。なお、「場所」のデフォルト値は、「ツール」->「オプ ション」->「プロジェクトおよびソリューション」->「全般」で変更できます。

すでに作成されている「ソリューション」の中に、新たな「プロジェクト」を追 加作成したい場合は、「場所」と②の欄の「ソリューション」名を指定し、「ソリュ ーションのディレクトリを作成」は行いません。

以下の図は、Visual Studio における「ソリューション」と「プロジェクト」の関係 を表したものです。「プロジェクト」は、ソースコードを管理する最小単位です。生 成されるプロジェクトファイルの拡張子は、pvfprojです。「ソリューション」とは、 最初のプロジェクトを作成すると自動的に生成されますが、複数のプロジェクトを 管理する単位です。例えば、プロジェクト間の依存関係等を管理します。一般的に、 一つの「ソリューション」の中に一つの「プロジェクト」の構成で使用することが 多いのですが、例えば、PVF Fortran コードと Visual C++コードから成る混合プロ グラムの場合は、明確に言語種別で「プロジェクト」を分けて構成しなければなり ません。



2.3 PVFプロジェクト・テンプレート

Visual Studio 上での PVF は、以下に示すプロジェクト・タイプのテンプレートを 用意しています。

ウィンドウを使わないキャラクタベースの入出力を伴うアプリケーションを作成 するためのプロジェクト。一般的には、これが多用されます。

■ Dynamic Library (ダイナミックライブラリ)

DLL モジュールを作成するためのプロジェクト。DLL とは、プログラムが必要と される時に、オンデマンドでローディングされるエグゼキュータブル・ファイルで す。

■ Static Library (スタティックライブラリ)

実行モジュールを生成する際に、リンクすることが可能な一つもしくは複数のオ ブジェクトから成るアーカイブ・ファイルを作成するためのプロジェクト。

Windows Application $(\dot{p} \prec \nu \dot{r})$

ウィンドウ、ダイアログ・ボックス、メニュー等のコンポーネントを使用する GUIを備えたアプリケーションを作成するためのプロジェクト。このようなアプリ ケーションのプログラム・エントリ・ポイントの名前は、WinMain となります。 ■ Empty Project (空のプロジェクト)

既存のアプリケーション (ソースコード等)を PVF に移行させる際に使用するスケルトン・プロジェクトです。これも、よく使用します。

2.4 新規にプロジェクトを作成し、新規にプログラム開発するための手続き

Visual Studio 上で新規にプロジェクトを作成して、その配下で新たにプログラム のコーディング並びに開発作業を行うための準備について説明します。一般には、 すでに所有している「プログラム・ソース・ファイル」を使用し、これを PVF プロ ジェクトに統合して開発を続行する方法がとられますが、これについては次項 2.5 項で説明します。

■ プロジェクトの新規作成

「ファイル」→「プロジェクト」を開き、「PGI Visual Fortran」のテンプレート の中の「Console Application」を選択します。なお、64 ビット Windows の場合は、 64 ビットアプリケーションあるいは、32 ビットアプリケーションの作成のどちら かを選択した上で、行ってください。

新しいプロジェクト						?	×
▶ 最近使用したファイル	.NET Fr	amework 4.5.2	 並べ替え: 既定 	-	📰 🔚 インストール済み テンプレート の検	ρ-	
 ▲ インストール済み ▲ テンプレート ▶ Visual C# ▶ Visual Basic Visual F# ▶ Visual C++ SQL Server ▲ PGI Visual Fortr × C44 ▶ インテル(R) Visua ▶ オンライン 	an al Fortran		Console Applicati Dynamic Library (Static Library (64- Windows Applica Empty Project (64 オンラインでテンプレー	ion (64-bit) (64-bit) -bit) -tion (64-bit) 4-bit)	PGI Visual Fortran PGI Visual Fortran PGI Visual Fortran PGI Visual Fortran PGI Visual Fortran	Type: PGI Visual Fortran An empty PGI Visual Fortran project	
名前(N): PVFProject4 場所(L): c:¥users¥kato¥do ソリューション名(M): PVFProject4		ocuments	¥visual studio 2015	¥Projects	•	参照(B) ソリューションのディレクトリを作成(D) ソース管理に追加(U) OK キャンセ)	ı،

■ PVFプロジェクト画面表示例

プロジェクトを新規作成した場合、ConsoleApp.f90 と言うファイル名で Fortran のスケルトン・コードが作成されます。この中でプログラムを開発します。

 PVFProject1 - Microsoft Visual Studio ファイル(E) 編集(E) 表示(Y) プロジェクト(E) ビルド(E) デバッ・ ヘルプ(H) 	$\mathcal{T}(\underline{\mathbb{D}}) = \mathcal{F} - \mathcal{L}(\underline{\mathbb{M}}) \operatorname{SQL}(\underline{\mathbb{Q}}) \mathcal{Y} - \mathcal{H}(\underline{\mathbb{T}})$	クイック起動 (Ctrl+Q) ク - ロ × テスト(<u>S</u>) 分析(<u>N</u>) ウィンドウ(<u>W</u>)
●・● 〒・■ ■ ■ P・C・ トアタッチ…・ Debug ・ ConsoleApp.f90 + × ConsoleApp.f90 + × ConsoleApp.f90 Fortran Console Application Generated by PGI Visual Fortran(R) 2013/02/25 21:01:38 Program prog implicit none ! Variables ! Body end program prog	〕⊯ ৣৣৣৣৣৣৣৣৣৣৣৣৣঢ়ঀৣৣ৾ঀ৵৸ ৸ ৸ ৸ ৸ ৸ D編集領域	
100 % ・ 《 出力 出力元の表示(<u>S</u>): ・ エラーー覧 出力	(1) 등 1 등 1 등 1 등 1 등 1 등 1 등 1 등 1 등 1 등	□-ド分析 ソリューション エクスプローラー プロ/ディ ・ ↓ × ・ ■ 型↓

「ソリューションエクスプローラ」は、IDE 管理の下にある「ソリューション」、「プ ロジェクト」、その配下の各ファイル・フォルダを管理するためのものです。この中 でソースファイル等が管理できます。

■ 新しいソースファイルを追加

「プロジェクト」内に新しいソースファイルを追加したい場合は、「プロジェクト」 → 「新しい項目の追加」を選択します。あるいは、「ソリューションエクスプロー



© SofTek Systems, Inc

ラ」内で、プロジェクト名の文字列部分を右クリックし、「追加」-> 「新しい項目」 を選択します。

Fortran のソースファイルのテンプレートを選択し、ファイル名を記述し、「追加」 ボタンをクリックします。これによって新しいファイル(SourceFile1.f90)が作成さ れます。以下の図では、テンプレートとして、Free-Format Fortran source file (.f90) を選択します。

新しい項目の追加 - PVFProject1		2
▲ インストール済み	並べ替え基準: 既定 🗸 🔡 🔚	インストール済み テンプレート の検乳 🔎 -
PGI Visual Fortran	Free-Format Fortran source fits (.f90) PGI	Visual Fortran 種類: PGI Visual Fortran
▶ オンライン	Fixed-Format Fortran source file (.f) PGI	A free-format Fortran source file Visual Fortran
	Fortran include file (.fi) PGI	Visual Fortran
	CUDA Fortran source file (.cuf) PGI	Visual Fortran
指定する	3	
名前(N): SourceFile1.f	90	
		追加(A) キャンセル

新しいソースファイルが作成されます。



2.5 既存のソースファイルを PVF プロジェクトに移行するための手続き

現在、Windows 上のフォルダに既存のプログラムファイルを有しており、これを Visual Studio の PVF プロジェクトに移行する方法を説明します。

■ プロジェクトの新規作成

「ファイル」 → 「プロジェクト」を開き、「PGI Visual Fortran」のテンプレート の中の「Empty Project」を選択します。さらに、ここでの例として、ソリューショ ン名を「FFTE2011」とし、プロジェクト名を「FFTE」と言う名前で定義します。



「Empty Project」を作成すると「ソリューションエクスプローラ」の中に、空のフォルダが作成されます。



■ 既存のソースファイルを P V F 環境へ移行

さて、これより、この IDE 環境の中に、既存のソースファイルを一つの「PVF プロジェクト」としてリンクします(移行します)。この実現方法にはいくつかの方法がありますが、ここでは、「ソリューションエクスプローラ」の中から操作する方法を説明します。「ソリューションエクスプローラ」の中の「Source Folder」を右クリックして現れるメニューの「追加」を選択し、「既存の項目(G)」を選びます。



次に、「追加」する既存のファイルを指定するための画面が現れます。 Shift キーを押しながら、"FFTE"プロジェクトに組み込みたいソースファイルを選択 し、「追加」ボタンを押します。

■ 既存項目の追加 - FFTE ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■								
〇〇-	ディスク(D:) 🕨 PGI 🔸 ffte-5.0 🖡	- - - - - - - - - -	ffte-5.0の検索	Q				
整理 ▼ 新しいフォルタ	:== -	- 1 0						
🔰 ダウンロード 🔺	名前	更新日時	種類	サイズ				
📃 デスクトップ	🏭 mpi	2013/02/25 20:50	ファイル フォル					
🗐 最近表示した場所	PVF	2013/02/26 11:15	ファイル フォル…					
	퉬 tests	2013/02/26 11:17	ファイル フォル…					
🍃 ライブラリ	F dzfft2d.f	2011/11/03 12:00	Fortran Source	4 KB				
■ ドキュメント	F dzfft3d.f	2011/11/03 12:00	Fortran Source	5 KB				
E	F fft235.f	2011/11/03 12:00	Fortran Source	7 KB				
■ ドデオ	F kernel.f	2011/11/03 12:00	Fortran Source	10 KB				
N Ra - Starb	F mfft235.f	2011/11/03 12:00	Fortran Source	8 KB				
	F vzfft1d.f	2011/11/03 12:00	Fortran Source	4 KB				
	vzfft2d.f	2011/11/03 12:00	Fortran Source	2 KB				
🜏 ホームグループ 🌱	F vzfft3d.f	2011/11/03 12:00	Fortran Source	3 KB				
	zdfft2d.f	2011/11/03 12:00	Fortran Source	4 KB				
👰 コンピューター	F zdfft3d.f	2011/11/03 12:00	Fortran Source	5 KB				
🚨 ローカル ディス	F zfft1d.f	2011/11/03 12:00	Fortran Source	4 KB				
- ローカルディス	F zfft2d.f	2011/11/03 12:00	Fortran Source	3 KB				
	F zfft3d.f	2011/11/03 12:00	Fortran Source	4 KB				
ファイ	ファイル名(<u>N</u>): "zfft3d.f" "fft235.f" "kernel.f" 🗸 🗸							
	追加(<u>A</u>)	キャンセル						

ソースファイルとヘッダーファイル(インクルードファイル)は、ファイルの種別 を確認して Visual Studio プロジェクト内の所定のファイル・フォルダの中に自動的 に登録されます。さらに、include ファイルがある場合は、同じ操作で、当該ファイ ルを登録します。(下図は、その様子を表しています)



正確に言うと、「既存」のソースファイルは、Visual Studio の当該プロジェクト用 のフォルダ内にコピーされるのではなく、既存のファイル・フォルダの位置を Visual Studio の PVF プロジェクトに登録すると言う形を取ります。したがって、Visual Studio (PVF)上で編集・変更されたソースファイル等は既存のフォルダ内のファイ ルが編集され、また、ファイルの削除を IDE 上で行った場合も、実際の既存フォル ダ内のファイルが削除されますのでご注意ください。しかし、既存のソースファイ ル以外のコンパイル時の中間オブジェクトファイルや生成される実行モジュールは、 Visual Studio でプロジェクトを登録した「場所」に保持されます。

■ **PVF**プロジェクト内にフォルダを新設

「ソリューションエクスプローラ」の中の「プロジェクト」配下には、予め用意されている三つのフォルダが存在します。その中の一つである「Source Files」フォルダ内に、(PVF に移行した) ソースファイルが登録されております。この「Source Files」フォルダ内にさらに新規のフォルダを新設し、その中にも既存のソースファイルを移行するための作業を行ってみます。「Source Files」フォルダの文字列を右クリックして現れるメニューの「追加」→「新しいフォルダ(D)」を選びます。以下の例のように新しいフォルダ (Newfolder1) が新設されますので、そのフォルダ名を設定します。

×	FFTE2011	Microsoft Visual S	Studio					クイック起動	(Ctrl+Q)	p -	. 🗆	x
ファイ	イル(E) 編	集(<u>E</u>) 表示(⊻)	プロジェクト(<u>P</u>)	ビルド(<u>B</u>)	デバッグ(<u>D</u>)	チーム(<u>M</u>)	SQL(Q)	ツール(I)	テスト(<u>S</u>)	分析(<u>N</u>)		
ウイン	ッドウ(<u>₩</u>)	ヘルプ(圧)										
6	- 🖸 🛅	- 🖆 🗎 🥙 🤊	- C - 🕨 アタ	'ツ ブ … ▼ D	ebug 👻 🎜	_} ⊨ @ *	p % ∎	위기적	Ŧ			
¢ z	fft3d.f	kernel.f	fft235.f ↔ 🗙			param.ł	1 1 × -	צ-בעע	ション エクス	プローラー ※	oo ▼ ₽	×
L – 7	0	FFTE: A FAST FO	DURIER TRANSFOR	I PACKAGE				006	5, •0 [™]	a > 🔉]	
:57	č	(C) COPYRIGHT S	SOFTWARE, 2000-	2004, 2008-	-2011, ALL RI	GHTS RESERVE	D		ション エクス	フローラーの	検家 (C	۰ م
Ŕ	Č	DAISUKE TAP	(AHASHI						ユーション 'F	FTE2011'(1)	プロジェ	クト)
I.	C C	FACULTY OF	ENGINEERING, II OF TSUKUBA	VFORMATION	AND SYSTEMS				FTE 3 Include File			
Ĭ	č	1-1-1 TENNO	DAI, TSUKUBA,	BARAKI 30	5-8573, JAPAN	l			n param.	h		
4	č	C-MAIL: Ual	SUKE@CS.(SUKUD)	a.ac.jp				_	Resource F	iles		
-14	C	ADIX-2, 3, 4, 5 AND 8 FFT ROUTINE						Source Files				
T. V.	Č,	EODTDANZZ SOUD							NewFol	der1		
λζ	č	FORTRANTT SOURC	JE PROGRAM						F fft235.f			
	C C	WRITTEN BY DAIS	SUKE TAKAHASHI						F Kernel.r			
		SUBROUTINE FFT2 IMPLICIT REAL*8 COMPLEX*16 A(*)	235(A,B,W,N,IP) 3 (A-H,O-Z) 2 B(*) W(*)						P 2ncoun			
1	.00 % - 4		,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,					コード分ね	所 ソリューシ	/ヨン エクスス	プローラ-	- 11
B	出力						• 4 ×	プロパティ	۲ ·····		∞ - - 7	×
1	出力元の表示	₹ <u>(S</u>): ピルド			• <u>6</u>	<u> </u>	ab⊒	IPVFPrjF	ProjHierFolde	erProps		-
								un set	ş			
								Folder	Name	NewFolder1		
								Unique	Identifier			•
							-	FolderNa	ame			
4	T=87	#5					•	Folder Na	ame			
-	- 」 見							_		_	_	_
この項	目はプレビュ	ューをサポートしてい	いません			21		55 列	55 文	?	挿	۵. ۸

ここでは、新しいフォルダを「tests」と定義し、前と同様に既存のソースファイル をこの「tests」フォルダの中へ移行・登録します。「tests」フォルダ上で右クリッ クして、「追加」→ 「既存の項目(G)」を選び、既存のソースファイル(以下の例で は speed3d.f)の場所を指定して登録します。この例では、この speed3d.f が Fortran のメイン・プログラムになります。



■ ルーチン間の依存性の解析について

プログラムが複数のルーチン、Fortran モジュール、インクルードファイルから構成され、さらにその関係に依存性がある場合は、そのコンパイルする順序が的確で

なければなりません。Linux の GNU 系の場合は、Makefile でこうした依存性を定義 しましたが、Visual Studio では、そのような Makefile を作成する必要がありません。 初回の「ソリューション・ビルド」において、ルーチン間の依存性の存在を検証し 内部的にコンパイルする順序の情報を保持します。

既存のソースファイルを PVF プロジェクトに移行した場合は、以下のように、「ビルド」 → 「ソリューションのビルド」を初回に行ってください。これによって、 ルーチン間の依存性解析を行い、その情報を保持します。



2.6 プログラムのコンパイルと実行(デバッグモード)

PVF上でプログラムをコンパイルする方法を説明します。Visual Studio でのデフ オルトのプロジェクト構成は、「デバッグ構成」のため、最適化レベルが0でシンボ リック情報が含まれた形で実行モジュールがビルドされます。

Ø	FFTE2011	- Microsoft Visual	Studio					クイック起動	(Ctrl+Q)	ρ -		×
ファ	イル(F) #	囑集(E) 表示(V)	プロジェクト(P)	ビルド(B)	デバッグ(D)	チーム(M)	SQL(Q)	ツール(T)	テスト(S)	分析(N)		
ウィ	ンドウ(W)	ヘルプ(H)										
	- 0		> - ♡ - ▶開始	- Debug	- 🔎 🗧	作やや	N 51	*N *N -				
4	zfft3d.f	kernel.f	fft235.f	Debug		speed3d.	f 🖮 🗙	- בעע	ション エクスご	プローラー …	а т ф	×
1	0	FETE: A FAST F	FOURTER TRANSFORM	(p)構成マオ	ページャー		-	÷ 001	`o - <i>ਵ</i> ੇ	ð Þ 🟳		
H.	H C								ション エクス	プローラー の枝	餘索 (C	۰ م
K	0	(C) COPYRIGHT BY	SUFTWARE, 2000-2	2004, 2008-	-2011, ALL RI	GHIS RESERVE	:U		7 - 2 - 2 'F	ETE2011' (1 🗆	r⊓≈:+/	, ДР)
L L	C DAISUKE TAKAHASHI C FACULTY OF ENGINEERING, INFORMATION AND SYSTEMS U UNIVERSITY OF TSUKUBA C 1-1-1 TENNODAI, TSUKUBA, IBARAKI 305-8573, JAPAN C E-MAIL: daisuke@cs.isukuba.ac.io C E-MAIL: daisuke@cs.isukuba.ac.io								FTE	1122011 (1)		~))
J.									Include File	25		
									🗅 param.ł	h		
_												_

■ ソリューションのクリーン

Visual Studio でソリューションをビルドする前に、クリーンアップします。「ビ ルド」→ 「ソリューションのクリーン」を実行します。

• • • • • • • • • • • • • • •		
FFTE2011 - Microsoft Visual Studio		クイック起動(Ctrl+Q) ク - ロ へ
ファイル(F) 編集(E) 表示(V) プロジェクト(P)	ビルド(B) デバッグ(D) チーム(M) SQL(Q) ツール(T) テスト(S) 分析(N)
ウィンドウ(W) ヘルプ(H)	当 ソリューションのビルド(B)	Ctrl+Shift+B
- C - C 🎦 - 🖆 💾 🧬 - C - 🕨 開始	ソリューションのリビルド(R)	
⊈ zfft3d f kernel f fft235 f	ソリューションのクリーン(C)	
	ソリューションでコード分析を実行(Y)	Alt+F11
C FFTE: A FAST FOURIER TRANSFOR	▲ FFTE のビルド(U)	
C (C) COPYRIGHT SOFTWARE, 2000-	FFTE のリビルド(E)	ン エクスプローラー の検索 (C ク・
C BY C DAISUKE TAKAHASHI	FFTE のクリーン(N)	-ション 'FFTE2011' (1 プロジェクト)
C FACULTY OF ENGINEERING, I	プロジェクトのみ(J)	E
C UNIVERSITY OF ISUKUBA	*************************************	nclude Files
C E-MAIL: daisuke@cs.tsukub		p paramin Pesource Files
	ハッチ ヒルト(1)	Bource Files
C ZFFI3D SPEED TEST PROGRAM	構成マネージャー(0)	1 tests
C FORTRAN77 SOURCE PROGRAM		F speed3d.f
C WRITTEN BY DAISUKE TAKAHASHI		F fft235.f
C IMPLICIT REAL*8 (A-H,O-Z)		F kernel.f
PARAMETER (NDA=16777216)		F zfft3d.f
100 % = 4		▼ コード分析 ソリューション エクスプローラー
100 % • •		
	* *	
出刀元の表示(5): ヒルト		IPVFPrjProjHierFolderProps +
	↑要、0 スキップ	
		FolderName tests
		UniqueIdentifier
		FolderName
エラーー覧 出力	•	rolder Name
ビルド正常終了		

■ ソリューションのビルド

次に、「ビルド」 → 「ソリューションのビルド」を実行します。ビルドの出力ロ グが表示されます。「FFTE build succeeded」と表示されますとビルドが成功した ことを意味します。



■ プログラムの実行 (デバッグなし)

ビルドされた実行モジュールを実行します。「デバッグ」 → 「デバッグなしで開 始」を実行するとプログラムの実行が開始されます。

~	EETE201	1 - Microsoft Visual Si	tudio						カイッカ記動	(Ctrl±0)	0	-	
2		eter(F) 表示(V)	プロミナクト(P)	ビル.K(B)	デバ	`w <i>/</i> 7(D)	∓ − (, (M)	SOL (0)	ツール(T)	= 7 b(s)	↔析(N	0	
	インドウ(W)	ヘルプ(日)	/	C) (1 (0)	11	ウィンド	ייי) ב <u>ג</u> (w)	595(4)	2 70(1)	7 (1 (3)	22-01(14	·)	
		ം മലം.ജിത	- (* -) 問始	* Debug		グラフィ	ックス						
С. С.				Debug		デバッグ	副始(S)			F5			
Į	zfft3d.f	kernel.f	fft235.f		•	デバッグ	₩/L(C) ≳しで開始(H))		Ctrl+E5			γŦΧ
1	C C	FFTE: A FAST FO	URIER TRANSFORM	PACKAGE	im.	パフォー	マンス分析の間	/ 即始(A)		Alt+F2	ľ	Γ	
5	C C	(C) COPYRIGHT S	OFTWARE, 2000-2	004, 2008-		一時停止	たパフォー:	マンス分析の	D開始(Y)	Ctrl+Alt+	F2	- の検索	(C 🖓 -
Ъ	Ó		AHASHT			プロセス	こアタッチ(P)				-	(1プロ:	ジェクト)
	č	FACULTY OF	ENGINEERING, IN	FORMATION	~	例外(X)				Ctrl+Alt+I	e		
UI I	C C	1-1-1 TENNO	OF ISUKUBA DAI, TSUKUBA, I	BARAKI 305	6.	ステップ	イン(1)			F11			
K	0	E-MAIL: dai:	suke@cs∙tsukuba	.ac.jp	6	ステップ	オーバー(0)			F10			
Ę	Č.	7FETSD SDEED TE	ST DDOCDAM		3	ブレーク	// /(()/ ポイントの型3	〒(参211分(の))		50			
ポッ	č		SI PROGRAM			ブレーク	ポイントの使い	E/#≄⊮≂(G) ∜(R)		F9			
К К	0	FURTRAN77 SOURC	E PROGRAM		29	すべての	ブレークポイ	、(D) ントの削除(Ctrl+Shift	+F9		
	0	WRITTEN BY DAIS	UKE TAKAHASHI		Ŭ.,	オペアの	データレント	をわいマ(A)	- /				
		IMPLICIT REAL*8	(A-H,O-Z)				>	±シシア(A) ポート(Y)					
		COMPLEX*16 A (ND	A)			データビ	ントのインポ-	– K(T)					
	100 % *	4				+++====	シャシャライン (の)	1 (1)				-םכגי	-9-
	出力				6								Ψ×
	出力元の表	眎(S): ビルド			-	FFILOU							-
	FFTE bui	d succeeded.						4		p			
	Build log	; was saved at " <mark>file:</mark>	//D:¥PGI¥ffte-5.0	¥tests¥PVF¥F	FTE20	11¥FFTE¥×6	4¥Debug¥Bu i I	dLos.htm*					
		= すべてリビルド: 1	正常終了、0 失敗、	0 スキップ									
	4							`					
	エラー一覧	[出力											
準備	完了						1	5 行	11 列	11 文	字		挿入
C:	C:¥W	NDOWS¥syst	em32¥CMI).EXE								_ 0	×
	_	F 1											
ľk	kato	lest											
	NX,NY	,NZ =											
10	10,100	,100											
	NX =	10	10 NY =			100	NZ =		100	TIME			
	0.224	41065586404	147	497	7.9	77393	776615	6	MFLO	PS			
FC)RTRAN	STOP											
100	-/+	にけ何かま	ーを押し	てくだえ	±1	١.							
結	ਯੋਜ ਯੋ 🗠				_								
続	泊9 乞												-1
続	ਮਿਸ9 ਟ 												-

■ プログラムの実行 (デバッグあり)

ソースレベルでデバッグを行いたい場合は、予め、プログラムの表示ウィンドウ 内で、「ブレークポイント」を設定しておきます。「ブレークポイント」の設定は、 対象となるソースラインの一番左端をクリックすることで設定できます。デバッグ 付で実行した場合、このブレークポイントで実行が停止します。

<u>プログラムのソース行番号を付けて表示したい場合</u>

Visual Studio の「ツール」→「オプション」ダイアログを開き、「テキストエディ ター」→「Fortran」→「全般」を選んだ画面で、「行番号」にチェックを付けて OK ボタンを押して下さい。行番号がプログラム文の左端に現れます。

FFTE2011 - Microsoft Visual Studio	イック記動 (Ctrl+O) クーロ×
- ● ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) SOL(O)	ツール(T) テスト(S) 分析(N)
ウィンドウ(<u>W</u>) ヘルプ(<u>H</u>)	
○ • ○ 1 記 • 留 単 ジ • ペ • ▶ 開始 • Debug • 月 _→ 目 幅 マ や ■ 対 対	×ii _∓
	ソリューション エクスプローラー 🚥 👻 🕇 🗙
C WRITTEN BY DAISUKE TAKAHASHI	ෙ ා රු ලෙ - අ ම 🕨 💭
H C IMPLICIT REAL*8 (A-H.O-Z)	ソリューション エクスプローラー の検索 (C 👂 🗸
PARAMETER (NDA=16777216)	👦 ソリューション 'FFTE2011' (1 プロジェクト)
DIMENSION LNX (3), LNY (3), LNZ (3)	▲ F FFTE
	 Include Files D. param.h
READ (5,*) NX,NY,NZ =	Resource Files
CALL FACTOR (NX,LNX)	🖌 🚔 Source Files
CALL FACTOR (NZ, LNZ)	🖌 🖬 tests
CALL INIT(A,NX*NY*NZ)	F speed3d.r
CALL ZFFT3D(A,NX,NY,NZ,U) CALL ZFFT3D(A,NX,NY,NZ,-1)	F kernel.f
LOOP=1 C	F zfft3d.f
IS 10 CONTINUE	コード分析 ハルコーション・エクスプローニー
100 % - 4	
	フロバティ * ¥ ×
出力元の表示(S): ビルド ・ 全 雪 香 荃 譚	· ·
A	
· · · · · · · · · · · · · · · · · · ·	
4	
エラー一覧出力	
準備完了 31 行	1列 1文字 挿入 📄

次に、「デバッグ」→「デバッグ開始」を実行するとプログラムのデバッグ実行が 開始されます。そして、最初の「ブレークポイント」で実行が停止します。



デバッグが開始されますと、「ブレークポイント」での変数値、配列値の確認を行い、 ステップ実行等の操作で問題となる部分の検証を行います。また、変数値の表示は、 ソースプログラム上の「変数」にカーソルを乗せると、その値が画面に表示されま す。



2.7 プログラムのコンパイルと実行(最適化オプションの適用)

PVF コンパイラによる「最適化オプション」を適用してビルドする方法を説明し ます。

■ ソリューションの最適化ビルド

最も簡単に行う方法は、ビルド・モードを「Debug」から「Release」に変更し てビルドすることです。(下図)



■ 新しい最適化ビルド・モードの作成

以下の操作は、一般的にほとんど行うことはないですが、新しく個人用のビルド・ モードを新設する際に以下のような方法をとります。「ビルド」 → 「構成マネージ ャ」を実行します。



構成マネージャの画面が出ましたら、「アクティブソリューション構成」のメニューを「Debug」から「<新規作成>」に変更します。

構成マネージャー				-	? 🗾	x
アクティブ ソリューション構成(<u>C</u>):	アク	ティブ ソリュー	->=	ョン プラットフォ	— <u>/</u> _(<u>Р</u>):	
Release	- x64					-
Debug Release	クト構	誠をチェック)(<u>R</u>):			
<新規作成>	プラ	ットフォーム		ビルド	配置	
<編集>	x64	[•			
<新規作成>を選択						
					閉じる	

新しいソリューション構成のウイザードが現れますので、この中に新しい構成の名 前を定義します。以下の例では、GPU コンパイル用の構成のために「Release-GPU」 としたものです。基本設定のコピー元はデフォルトの「Release」としています。

新しいソリューション構成 ?
名前(<u>N</u>):
Release-GPU
設定のコピー元(<u>S</u>):
Release 🗸
▼新しいプロジェクト構成を作成する(C)
OK キャンセル

次に、「プロジェクト」 -> 「…のプロパティ」を選択し実行します。



FFTE プロパ	ティ ページ	-			? ×
構成(<u>C</u>)	アクティブ(Release-GPU	J) 🔻	プラットフォーム(<u>P</u>):	アクティブ(x64)	▼ 構成マネージャー(<u>0</u>)…
▲ 構成プ	プロパティ 🔺	Displa	y Startup Banner	No	•
構成	t友 Release-(SPU (7	Include Directorie		
			ath	\$(IntDir)¥	
⊿ Fo	rtran	Objec	t File Name	\$(IntDir)¥	
	General	Debug	g Information Format	Disabled	
	Optimization	Optim	nization	Maximize Speed (-fastsse)	
	Preprocessing				
	Code Generation				
	Language				
	Floating Point Opt				
	External Procedur =				
	Libraries				
	Target Processors	Dicplay	Startun Pannor		
	Disgreaties	Display t	he compiler's startun k	apper (_V)	
	Diagnostics	Display a	ie complier s startup i	dinier (-v).	
	Proming				
	Command Line				
Lir	ker				
	sources				
⊳ Bu	ild Events				
<	TII F				
				ОК	Fヤンセル 適用(<u>A</u>)

プロジェクトの構成プロパティの画面となっていますので、その中の「Fortran」 と「Linker」のオプション項目の中で必要なものを指定します。ちなみに、上記で 定義した「Release-GPU」の基本設定のコピー元は、「Release」であるため、これ はデフォルトで「最適化オプションが設定済み」のモードのため、最適化オプショ ンはすでに適用済みと考えてください。その他の追加のオプションを任意で指定し ます。以下の図は、「Optimization」のオプション欄の例を示しています。

FFTE プロパティ ページ		?
構成(C): Release-GPU	 プラットフォーム(P): アクティブ(x64) 	▼ 構成マネージャー(0)
構成(C): Release-GPU ・構成プロパティ General Debugging Fortran General Optimization Preprocessing Code Generation Language Floating Point Opt External Procedur Libraries Target Processore Target Accelerato	 ブラットフォーム(P): アクティブ(x64) Optimization Maximize Speed (-fast) Global Optimizations Default Inlining No Use Frame Pointer No Loop Unroll Count Default Auto-Parallelization No Maximum Speed (となっているはず Vectorization Colorization	 ● 構成マネージャー(0) ■ -fast)
Diagnostics Profiling Runtime Command Line Linker Resources Build Events Cuctors Build Events	Select vectorization options (-Mivect, -Mivect=sse, -Mivect=sima, -	MVec キャンセル 適用(A)

追加するオプションとして、GPU 対応の OpenACC オプション (Enable OpenACC Directives) を Yes とします。

FFTE プロパティ ページ		S ×
構成(C): アクティブ(Release-GP	IJ) → プラットフォーム(P): アクティブ(x64)	→ 構成マネージャー(0)
構成(C): アクティブ(Release-GF 構成プロパティ General Debugging Fortran General Optimization Preprocessing Code Generation Language Floating Point Opt External Procedur Libraries Target Processors Target Processors Target Accelerato Diagnostics Profiling Runtime Command Line Linker	J) $7 \exists y \land 7 \exists - \Delta(P)$: $p \not p \exists \tau \exists (x 64)$ Fortran Dialect Fortran 95 Treat Backslash as Characte Yes Extend Line Length No Enable OpenACC Directives Yes MP1 None Enable CUDA Fortran No Enable OpenACC Directives Fortran Enable OpenACC Directives Enable OpenACC Directives Enable OpenACC Directives Enable OpenACC Directives	 ▼ 構成マネージャー(0)
Resources Build Events Gustam Build Chan		
		OK キャンセル 適用(A)

コンパイルオプション、リンクオプションを上記のような方法で設定する以外に、 さらに、「Fortran」→「Command Line」にて、その他の最適化オプションを直に指 定することも可能です。同様に、「Linker」→「Command Line」にも指定が可能で す。

こうして、いくつかの追加オプションを定義することにより「Release-GPU」構成のビルド・モードが定義されます。

■ プログラムの実行(最適化オプションあり)

ビルドされた実行モジュールを実行します。「デバッグ」 **→** 「デバッグなしで開 始」を実行すると最適化されたプログラムの実行が開始されます。



■ 並列化最適化オプションについて

① 自動並列化オプション

ソリューションエクスプローラ内の「プロジェクト名」をドラックします。これ は、このプロジェクトの「プロパティ」について変更するための初期操作となりま す。「プロジェクト」-「プロパティ」を選び、プロパティ画面を出します。

Himeno2 - Microsoft Visual Studio	クイック起動 (Ctrl+Q) 🛛 📮 🗖 🗙
ファイル(E) 編集(E) 表示(⊻) プロジェクト(<u>P</u>) ビルド(<u>B</u>) デバッグ(<u>D</u>) チーム(<u>M</u>) SQL(<u>Q</u>) ツー	·ル(<u>T</u>) テスト(<u>S</u>) 分析(<u>N</u>) ウィンドウ(<u>W</u>)
○ - ○ 語 - 留 🔛 🖓 - C - ▶ 795 - · · Release - β ₊ = ○ 10 → G.	६ ৫ % _₽ề 늘 때 한 한 ▮ 제 제 제 ₽
himenoBMTxp.f90 + ×	▼ ソリューション エクスプローラー ※※ ▼ 単 ×
integer :: nn	[‡] େଠୁରୁ ଡ୍-ଟ୍ଟି ≁ 💭
H integer :: ic, icr, icm real(4) :: flon ymflone? score yosa コプロミナクトタ	 ソリューション エクスプローラー の検索 (C ♪・
	□ <u>ソリューミュン 'Himono'' (1 プロミェクト)</u>
<pre>! ttarget specifys the measuring period in sec real(4).parameter :: ttarget=60.0</pre>	▲ F Himeno2
u real(8),external :: second	2 Include Files
call readparam	Resource Files
I Initializing matrixes	 Source Files
call initmem	F himenoBMTxp.f90
call initmt print *,' mimax='.mimax,' mimax='.mimax,' mkmax='.mkmax	
<pre>print *,' imax=',imax,' jmax=',jmax,' kmax=',kmax</pre>	
call system_clock(ic,icr,icm)	
dt=1.0/real(icr.8) print '(2x,a,e10.5)','Time measurement accuracy : ',dt !! Start measuring	
! 	
<pre>print *,' Start rehearsal measurement process.' print *,' Measure the performance in 3 times.'</pre>	_ コード分析 ソリューション エクスプローラー
100 % - 4	
出力	Himeno2 IPVFPrjProjHierProps
出力元の表示(S): ビルド	
	(Name) Himeno2
	ProiectPath D:¥PGI¥PVF¥Himeno ▼
	(Name)
۰ () () () () () () () () () (Name of the project
エラー一覧 出力	
この項目はプレビューをサポートしていません	الله.

「プロジェクトのプロパティ」画面で、「Fortran」 → 「Optimization」 → 「Auto-Parallelization」の設定を[Yes]とします。これによって、並列依存性のない μ ープ構造に対して、コンパイラが並列化を施します。

Himeno2 プロパティ ページ			? 🔀
構成(<u>C</u>): アクティブ(Release)	√ プラットフォーム(P):	アクティブ(x64) ・	構成マネージャー(<u>0</u>)…
▲ 構成プロパティ ▲	Optimization	Maximize Speed (-fastsse)	
General	Global Optimizations	Default	
Debugging	Vectorization	Default	
⊿ Fortran	Inlining	No	
General	Use Frame Pointer	No	
Optimization	Loop Unroll Count	Default	
Preprocessing	Auto-Parallelization	Yes	•
Code Generation			
Language			
Floating Point Opt			
External Procedur =			
Libraries			
Target Processors			
Target Accelerato	Auto-Parallelization		
Diagnostics	Enable auto-parallelization (-Me	concur).	
Profiling			
Runtime			
Command Line	-Mconcur を「Con	nmand Line I内に記述	ポしてもよい
⊳ Linker	C		
▷ Resources			
Build Events			
Custom Build Ston			
		ОК ‡ †	·ンセル 適用(<u>A</u>)

② OpenMP 並列化オプション

「プロジェクトのプロパティ」画面で、「Fortran」→「Language」→ 「Enable OpenMP Directives」の設定を[Yes]とします。これによって、コンパイラは OpenMP ディレクティブを解釈し、並列化コードを生成します。

Himeno2 プロパティ ページ	the second		?
構成(<u>C</u>): アクティブ(Release)	プラットフォーム(P):	アクティブ(x64) ・	構成マネージャー(<u>0</u>)
構成(C): アクティブ(Release) 構成プロパティ General Debugging Fortran General Optimization Preprocessing Code Generation Language Floating Point Opt External Procedur Libraries Target Processors Target Accelerato Diagnostics Profiling Runtime Command Line Linker Resources	 ブラットフォーム(P): Fortran Dialect Treat Backslash as Charact Extend Line Length Enable OpenMP Directives MPI Enable CUDA Fortran Enable OpenMP Directives Enable OpenMP Directives 	アクティブ(x64) Fortran 95 e Yes No Yes No None No Xtensions (-mp).	構成マネージャー(<u>0</u>)
Build Events Till Foundation			
		OK +7	ンセル 適用(A)

■ 自動並列、OpenMP 並列実行時の並列スレッド数の環境変数の設定

「Debug」あるいは「Release」のビルド・モードにおける「プロジェクトのプ ロパティ」画面で、「Debugging」→「Environment」の設定ボタンをクリックして、 以下の環境変数をセットします。これを事前にセットした後、プログラムを実行し てください。この変数を設定しなければ並列実行しません。

OMP_NUM_THREADS=<並列 CPU コア数> (例:OMP_NUM_THREADS=2) あるいは、

NCPUS=<並列 CPU コア数> (例:NCPUS=2)



あるいは、別の方法として、Windows®システム上の「環境変数」を設定する方法 があります。Windows®の「環境変数」の設定方法は、以下の URL をご参照ください。

なお、設定する変数は、上記で示した OMP_NUM_THREADS あるいは、NCPUS となります。この変数を反映させるために、本変数設定後、Visual Studio を起動す るようにしてください。

http://www.softek.co.jp/SPG/Pgi/win64/win64use.html

PVF コマンドプロンプト(下記、4章を参照)上で、各種の環境変数を指定する場合は、DOS コマンドの set コマンドを使用します。 以下は一例です。コマンドで 指定した場合は、そのコマンド画面上のみに有効です。

\$ set OMP_NUM_THREADS=2

2.8 プログラムの実行(入力データファイルのリダイレクト)

実行時に標準入力ファイルを指定して実行する方法を説明します。いわゆる、入 カデータを実行モジュールにリダイレクトする方法です。

■ 標準入力ファイルを指定する

「プロジェクト」 -> 「(プロジェクト名) プロパティ」を選択し、プロパティペ ージを開きます。



次に、「Debugging」→ 「Application Arguments」の欄に、"< ファイルパス名"と

表記する。"<" マークは、リダイレクトを意味し、これに続けてファイル名を記し ます。一方、アプリケーション実行のデフォルトのワーキング・ディレクトリは、 Visual Studio の「プロジェクトファイル」を含むディレクトリであるので、標準入 カファイルは、ここに置くだけでもよいです。

matmul プロパティ ページ	2 ×
構成(<u>C</u>): Release	 ・ プラットフォーム(P): アクティブ(x64) ・ ・ 構成マネージャ(Q)… ・ ・ ・
構成プロパティ General Debugging Fortran General Optimization Preprocessing Code Generation Language Floating Point Optio External Procedures Target Processors Target Accelerators Diagnostics Profiling Command Line Linker General Input Command Line Resources Puild Fuente III ・ *	Application Command Application Arguments Environment Merge Environment Working Directory *(SolutionDir) *(く C:¥tmp¥input_data" と言う風な記述 を行う。絶対パス名で指定しても良い。
	OK キャンセル 適用(<u>A</u>)

■ ワーキング・ディレクトリ(実行作業場所)の変更

「プロジェクト」→「(プロジェクト名)プロパティ」を選択し、プロパティペ ージを開きます。「Debugging」→「Working Directory」の欄に、実行時のワーキ ング・ディレクトリのパス名を指定します。デフォルトのワーキング・ディレクト リは、Visual Studio 上の使用プロジェクトの「プロジェクトファイル (***.pvfproj)」 が置かれているフォルダとなります。

このデフォルトのワーキング・ディレクトリを明示的に変更する際に指定します。 この変更を行うと、このディレクトリ・フォルダの中に入力データ等を置くことが できます

くじみり。			
matmul プロパティ ページ	A CONTRACT OF A DATE OF A	A R. D. R. B. BRELL, M. M.	? ×
構成(C): アクティブ(Debug)	・ プラットフォーム(P):	アクティブ(x64) ・	構成マネージャ(0)
構成プロパティ General Debugging Forban Linker Resources Build Events Custom Build Step	Application Command Application Arguments Environment Merge Environment Merge Environment Working Directory Working Directory Specify the application's working	NCPUS=2 Yes Created C:Ymp	
		ОК ‡ т	ンセル 適用(A)

2.9 MPI プログラムのビルド

■ MPI プログラムのプロジェクトの作成

MPIプログラムのプロジェクトを作成します。

MPIHELLO - Microsoft Visual Studio	クイック起動 (Ctrl+Q) 👂 🗕 🗖 🗙
ファイル(E) 編集(E) 表示(Y) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) SQL(Q) ツール	ル(<u>T</u>) テスト(<u>S</u>) 分析(<u>N</u>) ウィンドウ(<u>W</u>)
ヘルプ(圧)	
○ · ○ 昭 · 留 目 単 ヴ · ペ · ▶ 開始 · Debug · 月 _→ Ⅱ = ○ 「 ○ → ら. G, C	※, 늘 @ ▷ ▷ ♥ ■ 회 체 제
[↓] mpihello.f → ×	ソリューションエクスプローラー **** * * X
program hello include 'mpif.h'	G O 🏠 To - 2 🗇 🔑 🗖
H integer ierr, myid Sy call mpi init(ierr)	ソリューション エクスプローラー の検索 (C 👂 🔹
call mpi_cohm_rank(MPI_COMM_WORLD, myid, ierr) write(6.100) myid	👦 ソリューション 'MPIHELLO' (1 プロジェクト
100 format(1x,"hello - I am process",i3)	F MPIHELLO Traduda Files
I end	Resource Files
	🖌 🛁 Source Files
, , , , , , , , , , , , , , , , , , ,	F mpihello.f
X	
100 % · · · · · · · · · · · · · · · · · ·	ノロバテイ ▼ ¥ X
	<u> </u>
エラー一覧 出力	
準備完了 5 行	18 列 18 文字 挿入

■ プロジェクトのプロパティの設定

MPI プログラムのプロジェクトのプロパティに、MPI プログラムのコンパイル& リンクのための設定を行います。「プロジェクト」-「プロパティ」を開き、「Fortran」



- 「Language」を表示します。「MPI」の選択時に、「Microsoft MPI」を選択します。 このオプションの設定により、MPI プログラムのコンパイルとリンクが可能となり ます。具体的には、Microsoft MPI (MS-MPI)のインクルードファイルを取り込み、 MS-MPI ライブラリをリンクする設定を行います。

MPI プログラムの実行

次に、並列実行のための設定を行います。プロジェクトのプロパティ画面の [構成プロパティ] - [Debugging] に、MPI 実行の環境に関して設定を行うパラメータがあります。以下の画面イメージを見て下さい。「MPI Debugging」という項目のプル ダウンメニューで「Local」を選択して下さい。これは、このマシン上1台の中で MPI プロセスを起動して実行すると言う意味です。

MPIHELLO プロパティ ページ	100,000 PT. 100,000.		? ×
構成(C): アクティブ(Debug)	▼ プラットフォーム(P):	アクティブ(x64) ▼	構成マネージャー(0)
構成プロパティ General Debugging Fortran General Optimization Preprocessing Code Generation Language Floating Point Opt External Procedur Libraries Target Procesors Target Accelerato Diagnostics Profiling Runtime Command Line Linker Resources Build Events	 フラットフォーム(P): Application Command Application Arguments Environment Merge Environment MPI Debugging Working Directory MPI Debugging Enable MPI debugging, either log 	アクティブ(x64) ・ Yes Local Disabled Local	構成マネージャー(0) ▼
		OK +7	ンセル 適用(A)

その後、「適用」ボタンを押すと、さらに詳しいパラメータ項目が追加表示されま す。「Number of Processes」と言う欄は、並列プロセス数を指定する項目です。こ こに、並列実行に必要なプロセス数を指定します。なお、システム性能を考慮する と、システムのプロセッサが有する最大コア数が、この値に指定できる最大数です。 これらの設定は、Visual Studio の「Release」モードで行う場合は同様な設定を行 う必要があります。

この設定後に、Visual Studio 上で実行を行えば、自動的に mpiexec 実行が行わ れます。

Windows 環境上での MS-MPI の使用法の詳細に関しては、以下の URL をご覧く ださい。

http://www.softek.co.jp/SPG/Pgi/TIPS/public/general/msmpi_use.html

Himeno2 プロパティ ページ	CONTRACTOR OF	? 💌
構成(C): アクティブ(Debug)	√ プラットフォーム(P):	アクティブ(x64) 構成マネージャー(0)…
 構成プロパティ General Debugging Fortran Linker Resources Build Events Custom Build Step 	Application Command Application Arguments Environment Merge Environment Accelerator Profiling MPI Debugging Number of Processes Working Directory Additional Arguments: mpie Location of mpiexec Number of Processes Specify the number of processes	ACC_NOTYFY=1 Yes Local 4 * *SolutionDir 9 4 並列で実行するとした場合の 設定 es with which to run (mpiexec -n).
		OK キャンセル 適用(A)

3 GPU 用 OpenACC と CUDA Fortran を使用する

本章は、PGI アクセラレータ(Accelerator) コンパイラ・ライセンス(x64+GPU) の場合に使用できる機能を説明します。以下の画面イメージは、CUDA 5.5 をサポ ートした PVF 14.1 リビジョン以降の画面を使用しています。

3.1 OpenACC ディレクティブの利用

PGI アクセラレータ・プログラミングモデルは、ユーザがプログラム上にディレク ティブ(指示行)を挿入して GPU 並列計算ブロックを指示するモデルであり、プ ログラムの当該ループブロックを GPU上の kernel プログラムとして実行すること ができるようにコンパイラが翻訳することができます。予め OpenACC 用ディレク ティブを挿入してあるプログラムをコンパイルするための設定を説明します。

■ OpenACC を Enable にする設定

「プロジェクト」→「(プロジェクト名) プロパティ」を選択し、プロパティページを開きます。「Language」を選び、「Enable OpenACC Directives」を「Yes」にします(P21 参照)。これによって、ソースプログラム上の OpenACC のディレクティブを認識し、基本的にこの設定だけでも GPU 用のコード生成を行います。さらに、詳細な NVIDIA GPU のようなターゲット・アクセラレータの特性を指定したい場合は、以下の項目の設定も行うことが出来ます。

■ 「Target Accelerators」プロパティの設定

「プロジェクト」 \rightarrow 「(プロジェクト名) プロパティ」を選択し、プロパティページを開きます。「Target Accelerators」を選び、「Targeting NVIDIA Tesla」を「Yes」にします。これによって、Accelerator 用のディレクティブを認識し、GPU 用のコード生成を行います。「Targeting Multicore Host」はホスト上で OpenACC プログラムを並列実行させる executable を生成するオプションです。

Himeno-P	GI プロパティ ページ		7	? ×
構成(C):	Release	✓ ブラットフォーム(P): アクティブ(x64) ✓	構成マネージ	ัช−(О)
構成(C): ▲ 構成 G □ ▲ F	Release	✓ ブラットフォーム(P): アクティブ(x64) ✓ Target NVIDIA Tesla No Target Host No Target Multicore Host No Target NVIDIA Tesla Select NVIDIA Tesla accelerator target (-ta=tesla).	構成マネ-ジ ✓	·v-(0)
⊳ L ⊳ R <	Command Line inker lesources >	ОК + туt	د. د.الا	適用(A)

上記の通り、「Yes」にした後、細かな Accelerator 用のコンパイルオプションを表示させるには、<u>下部の「適用」ボタンをクリックすると、下図のような詳細なオプション・スイッチが現れます</u>。なお、PGI Accelerator 用のコンパイルオプションの詳細は、以下の URL にて説明していますので、ご覧下さい。

(PGI アクセラレータ・コンパイル用のオプション)

http://www.softek.co.jp/SPG/Pgi/TIPS/opt_accel.html

Himeno-P	GI プロパティ ページ							? X	
構成(C):	Release		✓ プラットフォーム(P):	アクティブ(x64)	~	構成マネーシ	νν−(O)	
▲ 構成 G D ▲ F	プロパティ ieneral bebugging ortran General Optimization Preprocessing Code Generation Language Floating Point Optic External Procedures Libraries Target Processors Target Accelerators Diagnostics Profiling Runtime Command Line inker	↑ Ta Se	Target NVIDIA Tesla Tesla Register Limit Tesla Use Fused Multiply-A Tesla Use Fast Math Library Tesla LLVM Tesla Noattach Tesla Noattach Tesla Autocollapse Tesla Debug Tesla Lineinfo Tesla Unroll Tesla Required Tesla Flush to Zero Tesla Flush to Zero	Yes No No No Defa Defa Defa Defa defa Defa target (-target (-target))	ult ult ult ult ult ult a=tesla).				
					ОК	キャンセ	μ –	適用(A)	

■ 「NVIDIA CUDA Toolkit」プロパティの設定

PVF にバンドルされている NVIDIA CUDA Toolkit のバージョンが表示されます。 複数の CUDA Toolkit のバージョンが現れた場合は、それを選択することができます。 なお、Toolkit とは別に、システムに実装されている NVIDIA のデバイス・ドライバ ーのバージョンを知りたい場合は、PVF Command Shell のコマンドプロンプト上 で pgaccelinfo コマンドを実行するとドライバーのバージョンが表示されます。

For a 8.0 driver: CUDA Driver Version 8000 と表示 For a 9.1 driver: CUDA Driver Version 9010 と表示 (一例)

t(C): Release	✓ プラットフォーム(P):	アクティブ(×64)	~ 相	豊成マネージャー(0)
構成プロパティ General Debugging ✓ Fortran General Optimization Preprocessing Code Generation Language Floating Point Optic External Procedures Libraries Target Processors Target Accelerators Diagnostics Profiling Runtime Command Line ▶ Resources	esla Pin Host Memory esla Autocollapse esla Debug esla Lineinfo esla Unroll esla Required esla Required esla Required esla Repuired esla Compute Capability esla Compute Capability esla Compute Capability esla Keep Kernel Files arget Host arget Multicore Host CUDA Toolkit fy the CUDA Toolkit com	No Default Default Default Default Default Default Default 7.5 8.0 9.0 9.1			

CUDA Toolkit のバージョンを明示的に指定して、その実行バイナリを作成する場合 は、上記の画面で示すとおり「NVIDIA CUDA Toolkit」のプロパティでそのバージョ ンを指定します。「Default」は、各 PVF バージョンによって異なります。

■ 「NVIDIA Compute Capability」プロパティの設定

((C): M	elease	✓ フラットフォーム(P):	アクティブ(x64) ~	構成マネージャー(C
構成プロ	パティ ヘ	Tesia Unroli	Delault	^
Gen	eral	Tesla Required	Default	
Deb	ugging	Tesla Flush to Zero	Default	
Fort	ran	Tesla Generate RDC	No	
0	General	Tesla CUDA Toolkit	8.0	
0	Optimization	Tesla Compute Capability	Manual	\sim
F	Preprocessing	Tesla CC Fermi	No	
(Code Generation	Tesla CC Kepler	No	
L	anguage	Tesla CC Maxwell	No	
F	Floating Point Optic	Tesla CC Pascal	No	
E	External Procedures	Tesla Keep Kernel Files	No	
1	Libraries	Target Host	No	
-	Target Processors	Target Multicore Host	No	
r	Disgnostics	-		×
	Profiling	Tesla Compute Capability		
, F	Runtime	Direct the compiler to automat	ically generate code compatible with all	applicable
	Command Line	compute capabilities of direct i	t to use a manually-selected set.	
⊳ Link	er			
, <u> </u>	~			

NVIDIA の GPU には、そのハードウェア特性を識別するために、「NVIDIA Compute Capability」(2.0, 2.1, 3.x. 5,x, 6.x)と言う番号が付けられており、PVF ではそのニックネームで表示しています。コンパイラはデフォルトで、自動的に CUDA Toolkit でサポートされる全ての Compute Capability 用のコードを生成しますが、明示的に Compute Capability を (複数) 指定してバイナリを生成することもできます。この 場合は、「Manual」にして「適用ボタン」を押し、画面に現れたボード種別を選択 してください。

■ PGI Accelerator 用のコンパイルメッセージ出力のプロパティの設定

PGI アクセラレータ用にコンパイルする際の詳細コンパイル情報を出力するには、 「Fortran」 → 「Diagnostic」 → 「Accelerator Information」を「Yes」とすることに より、コンパイル時、「出力ウィンドウ」内に詳細なコンパイル情報を出力すること ができます。

■ PGI Accelerator (OpenACC) 用の環境変数の設定

PGI アクセラレータ用の実行モジュールを実行する際の環境変数の設定は、以下の ように行います。「Debugging」→「Environment」の入力欄に、環境変数を入力し ます。以下の例は、ACC_NOTIFY と言う、GPU 内の kernel 実行が行われる毎にと の動作特性を出力するための機能ですが、それを有効に設定するものです。

また、複数の GPU ボードが実装されているシステムでは、ACC_DEVICE_NUM と 言う環境変数でその論理番号を設定することにより、実行に使用する GPU を指定 することができます。

PGI アクセラレータ用実行時の環境変数については、以下の URL にて説明をしておりますので、ご参照ください。

Himeno-PGI プロパティ ページ	?	×
構成(C): Release	✓ ブラットフォーム(P): アクティブ(x64)	-(0)
▲ 構成プロパティ へ General Debugging ▲ Fortran General Optimization Preprocessing Code Generation Language Floating Point Optic External Procedures Libraries Target Processors Target Processors Target Accelerators Diagnostics Profiling Runtime Command Line P Linker	Application Command Application Arguments Environment ACC_NOTIFY=1 Merge Environment Yes Accelerator Profiling Yes MPI Debugging Disabled Working Directory \$(SolutionDir) Environment Specify the environment for the application under debug, or variables to merge with the existing environment.	
< >>	OK キャンセル 適用	用(A)

http://www.softek.co.jp/SPG/Pgi/TIPS/opt_accel.html#ENV

■ PGI Accelerator (OpenACC) 用の簡易実行プロファイル出力の設定

OpenACC 用の実行モジュールの実行終了後、アクセラレータ上の実行プロファイル情報を出力するためのオプションがあります。「**Debugging**」->「Accelerator **Profiling**」を Yes とすることで有効となります。

PVF 入門ガイド

成(C): Release	✓ プラットフォーム(P): 1	アクティブ(x64) ~	構成マネージャー(O)
 構成プロパティ General Debugging Fortran General Optimization Preprocessing Code Generation Language Floating Point Optic External Procedures Libraries Target Processors Target Accelerators 	Application Command Application Arguments Environment Accelerator Profiling MPI Debugging Working Directory	ACC_NOTIFY=1 Yes Visabled \$(SolutionDir)	Y
Diagnostics Profiling Runtime Command Line > Linker > Resources	Accelerator Profiling Generate accelerator profiling in environment variable to 1.	formation at runtime. Sets the PGI_AC	C_TIME

3.2 PGI CUDA Fortran のコンパイル

PGI CUDA Fortran は、NVIDIA CUDA C と同じ考え方による GPU の処理を抽象化 できる Fortran 構文を実装し、NVIDIA GPU 用に直接プログラミングすることが可 能な Fortran コンパイラです。PGI Visual Fortran は、CUDA Fortran をフルサポー トし、CUDA Fortran コンパイル用のプロパティを設定することにより、コンパイ ルが可能となります。なお、CUDA Fortran のファイルは、.cuf と言うファイル拡 張子を付けることにより認識されますが、適切なリンケージ処理を行うために、以 下に説明する CUDA Fortran 用のプロパティを必ず設定して下さい。なお、PVF editor 上では、CUDA Fortran キーワード・構文の色識別ができます。



■ 「CUDA Fortran」プロパティの設定

「プロジェクト」 -> 「(プロジェクト名) プロパティ」を選択し、プロパティページを開きます。「Fortran」を選び、「Language」の中の「Enable CUDA Fortran」を「Yes」にします。これにより、CUDA Fortran 構文を認識してコンパイルを行うことができます。

Himeno-PGI プロパティ ページ		? ×
構成(C): Release	✓ ブラットフォーム(P): アクティブ(x64)	構成マネージャー(O)
▲ 構成プロパティ General Debugging ▲ Fortran General Optimization Preprocessing Code Generation Language Floating Point Optic External Procedures Libraries Target Processors Target Accelerators Diagnostics Profiling Runtime Command Line ▶ Resources	Fortran Dialect Fortran 95 Treat Backslash as Character Yes Extend Line Length No Enable OpenMP Directives No Enable OpenACC Directives No MPL None Enable CUDA Fortran Yes Enable CUDA Fortran Yes Enable CUDA Fortran Yes	
	ОК + +У	セル 適用(A)

なお、「Yes」にした後、細かな CUDA Fortran 用のコンパイルオプションを表示させるには、一旦、下部の「適用」ボタンをクリックすると、下図のような詳細なオプション・スイッチが現れます。

(PGI CUDA Fortran 用のオプションの説明 URL)

http://www.softek.co.jp/SPG/Pgi/TIPS/opt_cudaF.html

Himeno-PGI プロパティ ページ	? ×
構成(C): Release	 ブラットフォーム(P): アクラィブ(x64) 〜 構成マネージャー(O)
▲ 椹成プロパティ General Debugging ▲ Fortran General Optimization Preprocessing Code Generation Language Floating Point Optic External Procedures Libraries Target Processors Target Accelerators Diagnostics Profiling Runtime Command Line ▷ Linker ▷ Resources	Fortran Dialect Fortran 95 Treat Backslash as Character Yes Extend Line Length No Enable OpenMP Directives No Enable OpenACC Directives No MPI None Enable CUDA Fortran Yes CUDA Fortran Register Limit V CUDA Fortran Use Fused Multip Default V CUDA Fortran Debug Default CUDA Fortran Use Fast Math Lit No V CUDA Fortran Debug Default CUDA Fortran Ine Information Default V Enable CUDA Fortran V Enable CUDA Fortran V
	OK キャンセル 適用(A)

4 PVF コンパイラの起動(コマンド・ライン)

4.1 PVFコマンドプロンプトの起動

PVF Command Prompt(32bit) あるいは、PVF Command Prompt(64bit)のウィン ドウを開き、コマンドベースでコンパイラを操作することができます。PVF コマン ドプロンプトは、以下の方法で起動できます。

「スタート」メニューをクリック後、「すべてのプログラム」->「PGI Visual Fortran」 -> 「Command shells」-> 「PVF for VS 20xx Cmd」(64 ビット用)を選択すると、 コマンドプロンプト画面(ウィンドウ)が現れます。



このウィンドウのサイズや文字色、背景色等の「プロパティ」を変更するには、ウ ィンドウ上部(以下の例では、青地部分)にカーソルを置き、右クリックで下記の ようなプルダウンメニューが現れますので、この中の「プロパティ」で、カスタマ イズ・変更してください。なお、<u>プロパティを変更してその特性を保存したい場合</u> は、予め、「**管理者権限**」で「PVF for VS 2015 Cmd」を起動しなければなりません。

「PVF for VS 2015 Cmd」部分にカーソルを置いて右クリックすると、「管理者とし

com 管理者: PGI Visual Fortran 2008 (64)	- O X
	•
C:¥Users¥kato>dir	
ドライブ C のボリューム ラベルがありません サイス変更(S)	
ボリューム シリアル番号は 647A-68B9 です 🔷 ^{最小化(N)}	
□ 最大化(X)	
C:¥Users¥kato のディレクトリ	
×閉じる(C)	
2007/03/17 15:06 〈DIR〉 . 編集(E) 🔸	
2007/03/17 15:06 〈DIR〉 既定値(D)	
2010/06/07 17:34 8,461 .bash_h プロパティ(P)	
2007/03/27 17:20 175 .cshrc	
2007/03/14 17:28 <dir> Contacts</dir>	
2009/08/19 18:21 <dir> Desktop</dir>	
2008/07/26 20:14 <dir> Documents</dir>	
2009/08/04 14:01 <dir> Downloads</dir>	
2009/06/10 17:27 <dir> Favorites</dir>	
2007/03/14 17:28 <dir> LINKS</dir>	
2008/11/00 11:20 \DIRZ MUSIC	
2000/09/20 09:43 \DIR/ FICtures	
2007/03/14 17.20 \DIN/ 3aved dames	
2007/03/14 17:20 (DII// Searches	
2007/00/14 17:20 NDTV VIGEOS 2(個のファイル & 636 バイト	
13 個のディレクトリー13 780 353 024 バイトの空き領域	
C:¥Users¥kato>	-

て実行」がありますので、これを選んでコマンドプロンプトを起動してください。

4.2 PVFコンパイラ・コマンドの使用

PVF コンパイラの操作は、このコマンドプロンプト画面内のコマンド・ライン上 でテキストベースにより行います。コンパイラ・コマンドは、FORTRAN77 構文と その方言(IBM/DEC)のみを対象にした pgf77、並びに

FORTRAN77/Fortran90/Fortran95/Fortran2003 の構文を全て解釈可能な pgfortran (pgf90、pgf95 も同じもの)コマンドがあります。コマンドの使用方法に関しては、 「PGI Workstation & Server 製品」と同じであり、この詳細に関しましては、弊社 ホームページ上のコンテンツ、あるいは、ダウンロードサイトで提供しております 「PGI コンパイラ使用ガイド」(PDF ファイル)をご覧ください。なお、コマンド プロンプト画面内での Windows のコマンド体系は、DOS コマンドとなります。

(「PGI Workstation & Server 製品」でインタフェースとして提供している Linux の bash 環境と等価なものは、PVF ソフトウェアでは用意しておりません。別途、<u>PGI</u> Workstation ソフトウェアをインストールしてください)



```
一般に、コマンドプロンプト内でのコマンド使用の例を以下に記します。
Microsoft の DOS コマンドを使用します。
```

```
PGI Visual Fortran (64)
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:¥Users¥kato>cd C:¥ (ディレクトリを C:¥ トップへ)
C:¥>cd PGI (PGI と言うフォルダヘディレクトリ移動)
C:¥PGI>dir (フォルダ内のファイルリストを表示)
 ドライブ C のボリューム ラベルがありません。
ボリューム シリアル番号は 2863-1941 です
C:¥PGI のディレクトリ
2007/05/28 13:25 <DIR>
                         .
2007/05/28 13:25 <DIR>
                          . .
                       33 test.f
2007/05/10 14:53
          1 個のファイル
                                  33 バイト
           2 個のディレクトリ 45,879,316,480 バイトの空き領域
C:\PGI>pgf90 -fast -Minfo test.f (pgf90 コマンドを使用してコンパイル)
C:¥PGI>dir (コンパイル後のフォルダ内のファイルリストを表示)
 ドライブ C のボリューム ラベルがありません。
ボリューム シリアル番号は 2863-1941 です
C:¥PGI のディレクトリ
2007/05/28 13:25
             <DIR>
              <DIR>
                         •
2007/05/28 13:25
                          . .
                    24,576 test.dwf
2007/05/28 13:25
2007/05/28 13:25
                   143,360 test.exe
2007/05/10 14:53
                     33 test.f
               1,458 test.obj
2007/05/23 10:40
          4 個のファイル 169,427 バイト
           2 個のディレクトリ 45,879,316,480 バイトの空き領域
C:\PGI>test.exe (プログラムの実行)
hello!
```

(注意) コンパイル&リンク後に生成されるファイルは、*.exe ファイルと言う名称 の実行モジュールだけでなく、*.obj (中間オブジェクトファイル)、*.dwf (シンボル 情報ファイル) が生成されます。なお、*.dwf ファイルは、コンパイラが一時的に使 用するファイルですので、無視するかあるいは後で削除しても構いません。

■ 自動並列、OpenMP 並列実行時の並列スレッド数の環境変数の設定

コマンドプロンプト上で使用する場合、実行時に使用する様々な環境変数のセット の方法を説明します。Windows のコマンドプロンプト上での環境変数の設定は、 「set」コマンドで行います。これは、一般的な Windows 上でのルールと同じです ので、PGI コンパイラのランタイム時に指定する必要のある環境変数は、set コマ ンドでコマンドプロンプト画面を立ち上げる度に指定してください。これを事前に セットした後、プログラムを実行してください。 **\$ set OMP_NUM_THREADS=** {並列 CPU 物理コア数}

(例:set OMP_NUM_THREADS=2)

あるいは、

\$ set NCPUS= {並列 CPU コア数} (例: set NCPUS=2)

4.3 Windows®上で使用する際の留意点

Windows 上で PVF コンパイラをコマンドベースで使用する際の留意点は、以下の URL に補足説明をしております。基本的には Windows®のコマンド環境のルール をそのまま提供して結構です。

http://www.softek.co.jp/SPG/Pgi/win64/win64use.html

5 その他

5.1 実行モジュールの再配布

PVF コンパイラで生成された実行モジュールは、他の同種の Windows システム へ配布することができます。実行形式ファイルは、デフォルトでは静的リンク形式 なため、その実行形式のファイルのみのファイルで済みます。しかし、DLL 形式の 実行モジュールを作成した場合は、PGI 社が提供しているランタイム・ライブラリ である DLL(ダイナミック・リンク・ライブラリ)ファイルも併せて配布しなけれ ばなりません。この再配布可能な DLL ファイル群は、以下のディレクトリ配下にあ りますので、適時使用してください。基本的に、配布した実行モジュールと同じフォルダ内に必要な DLL が存在していれば、実行モジュールは動作します。

● 64 ビット Windows 上 C:¥Program Files¥PGI¥win64¥{リリース番号}¥REDIST (64bit モジュール用)

同様に、Microsoft Open Tools の再配布可能 DLL ファイルは、以下に存在します。 ● Microsoft Open Tools 用の DLL

C:¥Program Files¥PGI¥ Microsoft Open Tools 14¥redist(VS 2015 用)

5.2 PVF ドキュメント

PVF コンパイラのドキュメントは、Visual Studio の「ヘルプ」 内で、マニュア ルとして PDF ファイルを参照できます。

×	Himeno2 - Microsoft Visual Studio					5	マイック起動 (Ctrl-	+Q)	ρ = D	×
77	イル(F) 編集(E) 表示(V) プロジェクト(P)	ビルド(B)	デバッグ(D)	チーム(M)	SQL(Q)	ツール	(T) テスト(S)	分析(N)	ウィンドウ(W	1)
888	 ヘルプの表示(V) ヘルプロ表示(V) ヘルプコンテンツの追加と削除(C) ヘルプロシテンツの追加と削除(C) 	Ctrl+F1 Ctrl+Alt+F	}¥ , ∥	= 0 fa	⇒ ५. (a	答 。 15 何) ソリューション	ち や│ ∎ エクスプロ-	의 의 의 : 코- · · ·	Ψ×
<i>♀</i>	MSDN フォーラム(M) バグの報告(B)					+	C O G で Vリューション デュレット	9 - ₽ ๗ エクスブロ- •ヨン 'Himer	▶ 🔽 -ラー の検索 (0 102' (1 プロジェ	: _0 ~ :クト)
0	PGI Visual Fortran User's Guide PGI Visual Fortran Reference PGI Fortran Language Reference						▲ E Himen ■ Inde ■ Rese	o 2 ude Files ource Files		
	サンブル(S) カスタマー フィードバックのオブション(F) 副品の登録(P) トラブルシューティング(R)		, nknax				⊿ 🛀 Sou F h	rce Files himenoBMTX	(p.f90	
0	テクニカル サポート(T) DVD 版ヘルプの注文 オンラインのプライバシーに関する声明(O)		: ',dt							
-	print +, weasure the performance 100 % - 4	in a triles			_		コード分析 ソ! プロパティ	リューション	エクスプローラ	9- 9 ×
	出力 出力元の表示(5): ビルド		• 1	5 5 8	20 20	₽×				v
						Î				
	エラー一覧 出力									
進備	27				177		1列	1 文字	#	Eλ .

以 上