# SX-Aurora TSUBASAへの移植について

２０２０年　１１月　２０日

東北大学サイバーサイエンスセンター
日本電気株式会社

本資料は，東北大学サイバーサイエンスセンターとNECの共同により作成されたものです.

無断転載等は，ご遠慮下さい.

# 目　次

1. はじめに
2. SX-ACE と SX-Aurora TSUBASA の差異
3. NEC Numeric Library Collectionの利用
4. 移行の具体例
5. マニュアル情報

# はじめに

## 目的

- SX-ACEからSX-Aurora TSUBASAに移行にするにあたって
  プログラム移植の観点からの注意点を説明します。
- 移植時の事例と修正例を例示します。
- 説明は Fortranを前提とします。

# １．SX-ACE と SX-Aurora TSUBASAの差異

## システムの主な違い

# システムの主な違い

## システム構成

- SX-Aurora TSUBASA は，演算処理を行うベクトルエンジン（VE）と
  OS処理を行うベクトルホスト（VH）で構成されます。
- 1つのVHには、VEが8枚搭載されており、VE内では、8つのコアが48GBのメモリを共有
  しています。

| システム | 共有メモリ並列 | | 分散並列 | |
|---|---|---|---|---|
| | 最大並列数 | 最大使用メモリ量 | 最大並列数 | 最大使用メモリ量 |
| SX-Aurora TSUBASA | 8 | 48GB | 2,048 | 12,288GB |
| SX-ACE | 4 | 64GB | 4,096 | 65,536GB |

※ 最大並列数、最大使用メモリ量は、サイバーサイエンスセンターで1ジョブあたり利用可能な最大値

# 2. SX-ACE と SX-Aurora TSUBASAの差異

## プログラムの観点から

# プログラムから見た違い

## 主な違い

| | | SX-ACE | SX-Aurora TSUBASA |
|---|---|---|---|
| 1 | コマンド等 | ※1 | ※1 |
| 2 | エンディアン | ビックエンディアン | リトルエンディアン |
| 3 | ローカル変数 | bssセクションに割り付け | スタック領域に割り付け |
| 4 | HWの違い | ※2 | FMA演算器の採用 ※2 |

※1 コンパイラコマンド、コンパイラオプション、指示行、実行時環境変数も違います。
※2 演算器、ベクトルパイプライン数、等が違います。

次項から個別に説明します。

# コマンド等の違い

## 相違点 と 対処方法

| | | SX-ACE | SX-Aurora TSUBASA |
|---|---|---|---|
| 1 | コマンド等 | ※1 | ※1 |

※1 コンパイラコマンド、コンパイラオプション、指示行、実行時環境変数も違います。

- コンパイラコマンドの相違点

| | SX-ACE | SX-Aurora TSUBASA |
|---|---|---|
| Fortran | sxf90 | nfort |
| C/C++ | sxcc<br>sxc++ | ncc<br>nc++ |
| MPI | mpisxf90<br>mpisxcc<br>mpisxc++ | mpinfort<br>mpincc<br>mpinc++ |

# コマンド等の違い

## コンパイラオプション

- 主なコンパイラオプションの相違点

| 機能 | SX-ACE | SX-Aurora TSUBASA |
|---|---|---|
| 最適化レベル | -Chopt | -O3 |
| | -Cvopt | -O2 |
| 自動並列化 | -Pauto | -mparallel ※1 |
| OpenMP並列化 | -Popenmp | -fopenmp ※1 |
| 自動インライン展開 | -pi auto | -finline-functions |
| 自動インライン展開をしない | -Npi | -fno-inline-functions |
| バウンスチェック | -eR | -fbounds-check |
| ループアンロール | -O unroll | -flop-unroll |
| 編集リスト出力 | -L fmtlist | -report-format |
| プリプロセス | -Ep | -fpp |
| バージョン表示 | -V | --version (コンパイルしない) |

※1 自動並列とOpenMP並列の混在も可能

# コマンド等の違い

## コンパイラ指示行

- 指定方法

| 言語 | SX-ACE | SX-Aurora TSUBASA |
|---|---|---|
| Fortran | !CDIR | !NEC$　自由形式<br>*NEC$　固定形式 |
| C/C++ | #pragma | #pragma _NEC |

- 主なコンパイラ指示行の相違点

| 機能 | SX-ACE | SX-Aurora TSUBASA |
|---|---|---|
| 配列に依存関係がないことを指定 | nodep | ivdep |
| ループの入れ替えを行う | loopchg | interchange |
| ループの入れ替えを行わない | noloopchg | nointerchange |
| ループを展開する | expand | unroll_completely |
| N段アンローリングする | unroll=*n* | unroll(*n*) |
| 外側ループをn段アンローリングする | outerunroll=n | outerloop_unroll |
| 自動ベクトル化を許可しない | novector | novector |
| 配列をADBにバッファリングする | on_adb | なし |

- SX-Aurora TSUBASAでは、SX-ACE向け指示行をSX-Aurora TSUBASA向け指示行に変換するツールを提供しています（次頁で説明）。

# コマンド等の違い

## 指示行変換ツール

- Fortran: nfdirconv コマンド、 C/C++: ncdirconv コマンド
  - 指示行は 指示行変換ツールで一括変換が可能です。
  - 詳細は コンパイラユーザーズガイド の 付録C を参照下さい（マニュアルの詳細はPage 14を参照）。
  - 念のため、変換前後を確認し、
    さらに、コンパイル時に編集リストを採取して、意図した変換とベクトル化となっているかを
    確認することをお勧めします。

  - 代表的なオプション点

| **オプション** | **説明** |
|---|---|
| -a | sxf90/sxf03/sxcc /sxc++のコンパイラ指示行を削除せずに、<br>nfort/ncc /nc++のコンパイラ指示行を追加します。 |
| -p | sxf90/sxf03/sxcc /sxc++のコンパイラ指示行に対応する<br>nfort/ncc /nc++のコンパイラ指示行が無い場合、<br>sxf90/sxf03/sxcc /sxc++のコンパイラ指示行を削除しません。<br>(既定の動作) |
| -r | ディレクトリと同時に指定したとき、サブディレクトリを再帰的に走査します。<br>ディレクトリのシンボリックリンクは無視します。 |

  - 使用例）

```
$ nfdirconv sample.f
```

# 演習問題1

別紙「SX-Aurora TSUBASAへの移植について　演習問題」の「演習問題1. 指示行変換ツール」を実施します。

# コマンド等の違い

## 実行時環境変数

- 主な実行時環境変数の相違点

| 機能 | SX-ACE | SX-Aurora TSUBASA |
|---|---|---|
| 実行解析情報の出力 | F_PROGINF | VE_PROGINF |
| 入出力バッファサイズ指定 | F_SETBUF | VE_FORT_SETBUF |
| ファイル装置接続 | F_FFn | VE_FORTn |
| ビッグエンディアンモード指定 | F_UFMTENDIAN | VE_FORT_UFMTENDIAN |
| 自動並列数の指定 | F_RSVTASK | OMP_NUM_THREADS<br>VE_OMP_NUM_THREADS ※1 |
| トレースバック情報の出力 | F_TRACEBACK | VE_TRACEBACK |
| MPI 実行性能情報の出力 | MPIPROGINF | NMPI_PROGINF |
| MPI 通信情報の出力 | MPICOMMINF | NMPI_COMMINF |

※1 自動並列数とOpenMPスレッド数の指定方法は同じ

# コマンド等の違い

## マニュアル

- コマンド等の違いの詳細は以下のマニュアルに記載されています。
  - マニュアル掲載場所
    https://www.hpc.nec/documentation
  - Fortran Compiler ユーザーズガイド
    - コンパイラオプションの相違点　　付録B.2 Fortran90/SXコンパイラオプション
    - コンパイラ指示行の相違点　　付録C.3 コンパイラ指示行
    - 実行時環境変数の相違点　　付録B.4 環境変数
    - コンパイラ指示行変換ツール　　付録C. コンパイラ指示行変換ツール
  - NEC MPIユーザーズガイド
    - 実行時環境変数　　3.2.4 環境変数

# エンディアンの違い

## 相違点 と 対処方法

| | | SX-ACE | SX-Aurora TSUBASA |
|---|---|---|---|
| 2 | エンディアン | ビックエンディアン | リトルエンディアン |

- Fortranは実行時環境変数で対処可能、Cでは環境変数はありません(バイトスワップが必要)
- プリ/ポストツールとの連携に注意する必要があります
- 実行時環境変数　VE_FORT_UFMTENDIAN (Fortran) でビックエンディアンを扱えます
  使い方の一例

| 引数 | 説明 |
|---|---|
| ALL | export VE_FORT_UFMTENDIAN=ALL (sh形式の場合)<br>全ての装置番号をビックエンディアンとする |
| 番号,番号 | export VE_FORT_UFMTENDIAN=10,11（sh形式の場合）<br>装置番号10番と11番をビックエンディアンとする |
| 番号-番号<br>(連続の範囲) | export VE_FORT_UFMTENDIAN=10-12（sh形式の場合）<br>装置番号10番から12番をビックエンディアンとする |
| big:番号\|little:番号<br>と ; の除外 | export VE_FORT_UFMTENDIAN=big;little:10-12（sh形式の場合）<br>装置番号10番から12番**以外**をビックエンディアンとする<br>(;前の指定から、除外する装置番号を後ろに指定する) |

- SX-ACE で F_UFMTENDIAN (リトルエンディアン、Fortran) を指定していたものは設定不要です。

# ローカル変数の扱い方の違い

## 相違点 と 対処方法

| | | SX-ACE | SX-Aurora TSUBASA |
|---|---|---|---|
| 3 | ローカル変数 | bssセクションに割り付け | スタック領域に割り付け |

- bssセクション

  スタティックな領域。手続き実行でセットされた**値が以降も保持**されます。
  初期値なしスタティック領域であるが、
  実行に先立ち **値0(論理型はFALSE)に初期化**されます。

  ローカル(局所)変数であっても手続き終了後に、bss上で値が残っている。
  (save属性指定漏れでも、なんとかなってしまう場合がある)

- スタック領域

  save属性を指定しないローカル(局所)変数は手続きが終了後、**不定**になります。

  save属性漏れ、初期化漏れのローカル(局所)変数がある場合、
  SX-ACEの実行結果と結果が変わる、実行エラー等が発生する可能性がある

# ローカル変数の扱い方の違い

## 初期化漏れ等のソース修正

- 実行エラー、SX-ACEと結果が変わる場合はソース修正が必要となります。
  - ソースコードからsave属性漏れを探し、save属性を指定する
  - ソースコードから初期化漏れを探し、明示的な初期化を行う

ソース修正で結果が変わる場合がありますが、
プログラムとしては正しい記述、正しい挙動に近づきます。

実行時の例外検出方法
倍精度の場合
コンパイラオプション
　-minit-stack=nan
実行時環境変数
　VE_INIT_HEAP=NAN

- コンパイラオプションで対処する(次善の策)　(性能が低下する場合があります)

| **オプション** | **説明** |
|---|---|
| -bss | 局所変数を.bssセクションに割り付ける。 |
| -save | 各プログラム単位において(RECURSIVEであるものは例外とする)、SAVE文がすべての局所変数に指定されているものとする |
| -minit-stack=zero | 実行時にスタックに割り付ける領域を0(ゼロ)で初期化する |

-bss と-save オプションの違い
SAVE属性をもつ変数の値は、前回当該ルーチンが呼び出され、リターンしたときの値が、本呼び出し時の最初の値となりますが、-bssの場合はそれが保証されません。

- 実行時環境変数で対処する(次善の策)　(性能が低下する場合があります)

| **オプション** | **説明** |
|---|---|
| VE_INIT_HEAP=ZERO | 実行時にヒープに割り付ける領域を0(ゼロ)で初期化する |

0初期化をする場合はコンパイラオプション-minit-stack=zeroと併用することをお勧めします

# HWの違い

## 相違点 と 対処方法

| | | SX-ACE | SX-Aurora TSUBASA |
|---|---|---|---|
| 4 | HWの違い | ※2 | FMA演算器の採用 ※2 |

※2 演算器、ベクトルパイプライン等が違います。

- 演算器構成の違い
  - SX-Aurora TSUBASAではFMA演算器を採用
    SX-ACEとは丸め誤差レベルで演算結果が変わる可能性がある
  - 丸め誤差が気になる場合は コンパイラオプション -mno-vector-fma (ベクトル積和演算命令の使用を許可しない) でFMA演算を抑止することが可能
- 逆数近似処理のサポート
  - 除算/SQRT演算器をHW実装している。更に高速に処理するために逆数近似処理をサポートしている
  - 除算

| 説明 | コンパイラオプション | 誤差 |
|---|---|---|
| 除算器を利用 | -mvector-floating-divide-instruction | なし |
| 逆数近似を利用 **(既定値)** | -mno-vector-floating-divide-instruction | なし |
| 高速版逆数近似を利用 | -mvector-low-precise-divide-function | あり(仮数部に最大１ビット) |

実行速度は上から下の順で速くなる

# HWの違い

- SQRT

| 説明 | コンパイラオプション | 誤差 |
|---|---|---|
| SQRT演算器を利用 | -mvector-sqrt-instruction | なし |
| 逆数近似を利用 **(既定値)** | -mno-vector-sqrt-instruction | あり |

実行速度は上から下の順で速くなる

- ベクトル総和演算の計算結果について
  - 今までのSXシリーズと同様、ベクトル総和演算の計算結果に差分が生じる可能性あり（加算順序の違いが原因）
  - ベクトル総和演算を使用しない場合はコンパイラオプション -mno-vector-reduction で指定可能
    - 本オプションを使用するとベクトル化不可となるので、総和処理のコストが重い場合は実行時間が増大する

# 3. NEC Numeric Library Collectionの利用

## ASLのリンク方法、FFTW3インタフェース紹介

# NEC Numeric Library Collection 利用時の留意事項

## NEC Numeric Library Collection

- 前システムSX-ACEで提供されていた数値計算ライブラリASL、LAPACKなどの数学ライブラリの機能を包含したライブラリです。
- コンパイル、リンク時のオプションがSX-ACEの時と異なります。

| ライブラリ名 | | 機能概要 |
|---|---|---|
| ASL | ネイティブインタフェース | 数値計算・統計計算のための各種アルゴリズムを備えた科学技術計算ライブラリ |
| | 統合インタフェース | フーリエ変換、乱数、ソート |
| | FFTW3インタフェース | FFTW（Version 2.x）のAPIでASLのフーリエ変換を利用するためのインタフェースライブラリ |
| BLAS | | ベクトル、行列の基本演算 |
| LAPACK | | 連立一次方程式、固有値方程式、特異点分解 |
| ScaLAPACK | | 連立一次方程式、固有値方程式、特異点分解（分散メモリ並列用） |
| BLACS | | ベクトル、行列の基本演算のためのメッセージパッシングライブラリ（分散メモリ並列用） |
| SBLAS | | スパース行列の基本演算 |
| Hetero Solver | | 連立一次方程式（スパース行列用の直説法ソルバ） |
| Stencil Code Accelerator | | ステンシル計算の加速 |

# NEC Numeric Library Collection 利用時の留意事項

## コンパイル環境設定の実施

- プログラムのコンパイルを行う前に、コンパイル環境設定スクリプトを実行します。
- コンパイルするプログラムの分散並列の有無、および整数型のサイズによってスクリプト実行時の引数が異なります。

| 分散（MPI）並列版ライブラリの利用 | プログラムでの既定値整数型サイズ | スクリプト指定引数 |
|---|---|---|
| 無 | 32bit（kind=4） | （なし） |
| 無 | 64bit（kind=8） | i64 |
| 有 | 32bit（kind=4） | mpi |
| 有 | 64bit（kind=8） | mpi i64 |

例）64bit整数型を使用したMPI並列版ライブラリを使用する場合（sh形式の場合）

```
$ source /opt/nec/ve/nlc/2.1.0/bin/nlcvars.sh mpi  i64
```

## ジョブスクリプト内での設定

- 環境設定は、プログラム実行時に必要な環境変数も設定されるため、ジョブスクリプト内でも必要です。

# NEC Numeric Library Collection 利用時の留意事項

## コンパイルの実施

- 32bit整数型ライブラリ、64bit整数型ライブラリを利用するかでコンパイラオプションが異なります。
  (1) 32bit整数版ライブラリを使用する場合

| ライブラリ名 | | コンパイラオプション | リンクオプション |
|---|---|---|---|
| ASLネイティブインタフェース | 逐次 | なし | -lasl_sequential |
| | OpenMP並列 | -fopenmp | -lasl_openmp –fopenmp |
| ASL統合インタフェース | 逐次 | なし | -lasl_sequential |
| | OpenMP並列 | -openmp | -lasl_openmp –fopenmp |
| | MPI並列 | なし | -lasl_mpi_sequential |
| | ハイブリッド並列 | -openmp | -lasl_mpi_openmp –fopenmp |
| FFTW3インタフェース | 逐次 | なし | -laslfftw3 –lasl_sequential |
| | OpenMP並列 | -openmp | -laslfftw3 –lasl_openmp -fopenmp |
| | MPI並列 | なし | -laslfftw3_mpi –lasl_mpi_sequential |
| | ハイブリッド並列 | -fopenmp | -laslfftw3_mpi –lasl_mpi_openmp -fopenmp |

BLAS/LAPACKなど、上記以外のライブラリのコンパイラオプション、リンクオプションについては以下のマニュアルをご参照ください。
https://www.hpc.nec/documents/sdk/SDK_NLC/UsersGuide/main/ja/f_linking.html

# NEC Numeric Library Collection 利用時の留意事項

## コンパイルの実施

(2) 64bit整数版ライブラリを使用する場合

| ライブラリ名 | | コンパイラオプション | リンクオプション |
|---|---|---|---|
| ASL<br>ネイティブ<br>インタフェース | 逐次 | -fdefault-integer=8 | -lasl_sequential_i64 |
| | OpenMP並列 | -fdefault-integer=8<br>-fopenmp | -lasl_openmp_i64 –fopemp |
| ASL統合<br>インタフェース | 逐次 | -fdefault-integer=8 | -lasl_sequential_i64 |
| | OpenMP並列 | -fdefault-integer=8<br>-fopenmp | -lasl_openmp_i64 –fopemp |
| | MPI並列 | -fdefault-integer=8<br>-fdefault-real=8 | -fdefault-integer=8 –fdefault-real=8<br>-lasl_mpi_sequentiall_i64f |
| | ハイブリッド並列 | -fdefault-integer=8<br>-fdefault-real=8<br>-fopenmp | -fdefault-integer=8 –fdefault-real=8<br>-lasl_mpi_openmp_i64f –fopenmp |
| FFTW3<br>インタフェース | 逐次 | -fdefault-integer=8 | -laslfftw3_i64 –lasl_sequential_i64 |
| | OpenMP並列 | -fdefault-integer=8<br>-fopenmp | -laslfftw3_i64 –lasl_openmp_i64 –fopenmp |
| | MPI並列 | -fdefault-integer=8<br>-fdefault-real=8 | -fdefault-integer=8 –fdefault-real=8<br>-laslfftw3_mpi_i64f –lasl_mpi_sequential_i64f |
| | ハイブリッド並列 | -fdefault-integer=8<br>-fdefault-real=8<br>-fopenmp | -fdefault-integer=8 –fdefault-real=8<br>-laslfftw3_mpi_i64f –lasl_mpi_openmp_i64f<br>-fopenmp |

# NEC Numeric Library Collection 利用時の留意事項

## コンパイルの実施

- 使用例）
  - 逐次版ASLネイティブインタフェースを使用する場合

```
$ nfort sample.f –lasl_sequential
```

  - 64bit整数版逐次ASLネイティブインタフェースを使用する場合

```
$ nfort sample.f –lasl_sequential_i64
```

  - 64bit整数版OpenMP並列ASLネイティブインタフェースを使用する場合

```
$ nfort sample.f -fdefault-integer=8 –fopenmp -lasl_openmp_i64
```

  - 64bit整数版MPI並列統合インタフェースを使用する場合

```
$ mpinfort  sample.f -fdefault-integer=8 –fdefault-real=8 -lasl_mpi_sequentiall_i64f
```

# NEC Numeric Library Collection 利用時の留意事項

## FFTW3インタフェース利用時の留意事項

- 何点かの留意事項(配列の次元順、Cの複素数型 等)があるので利用前に参照ください。
- コンパイル、リンクは「コンパイルとリンク」ページを参照ください。
- ヘッダファイルの変更が必要です。

**FFTW3インタフェース (Fortran用)**

**概要**

FFTW3インタフェースは、FFTW version 3.xとほぼ互換のインタフェースをもつライブラリです (一部のルーチンおよび定数は未対応)。 FFTWのインクルードファイルを差し替えるだけで、FFTWを使用しているユーザプログラムから、 Vector Engine向けに高度にチューニングされたASLのフーリエ変換ルーチンを使用することができます。 オリジナルのFFTWは、FFTW公式サイト[1]で公開されているオープンソースのフーリエ変換ライブラリです。 詳細は、FFTWのドキュメント[2]を参照してください。

**FFTW3インタフェースの使用方法**

FFTW3インタフェースのインクルードファイルは、オリジナルのFFTWで提供されているものとファイル名が異なります。 FFTW3インタフェースを使用する場合は、以下のようにインクルードファイル名を変更してください。

Fortran 2003プログラムの場合

```
include "fftw3.f03"  ⇒  include "aslfftw3.f03"
```

Fortran 77プログラムの場合

```
include "fftw3.f"  ⇒  include "aslfftw3.f"
```

コンパイルとリンクには、FFTW3のコンパイル方法およびリンク方法を記載しています。

```
include "fftw3.f03"  ->  include "aslfftw3.f03"
include "fftw3.f"    ->  include "aslfftw3.f"
#include <fftw3.h>    ->  #include <aslfftw3.h>
```

- マニュアルページでサンプルコードの表示とダウンロードが可能です。

# 演習問題2

別紙「SX-Aurora TSUBASAへの移植について　演習問題」の「演習問題2. FFTW3インタフェース利用」を実施します。

# 4. 移行の具体例

# 移行の具体例

## 具体例を複数を提示します

- いずれもSX-ACEでは問題なく、コンパイル、実行できていた例です。
- SX-Aurora TSUBASA では コンパイルエラー、実行エラー、結果不正となり、ソース修正等で解消する例です。

# プリプロセッサの動作の違い

## プリプロセッサディレクティブでコンパイルエラーが発生する

- ソースファイルのサフィックスが .f90/.f の場合
  - SX-ACEとSX-Aurora TSUBASAで処理方法が異なる
    - SX-ACE　　#defineなどのプリプロセッサディレクティブがある場合、その行を無視してコンパイルされます。
    - SX-Aurora TSUBASA　　プロプロセッサディレクティブとして扱うため、コンパイルするにはコンパイラオプション **–fpp** を指定する必要があります。
- ソースファイルのサフィックスが .F90/.F の場合
  - SX-ACE、SX-Aurora TSUBASA ともにコンパイル時にプリプロセッサが常に動作します。

# 未サポートヘッダー

## 未サポートヘッダファイルのコメントアウト

- SX-Aurora TSUBASA向けコンパイル時に、記述を無効にする場合
  #ifndef __ve__ 、#endif で囲む

- 定義済みマクロ “__ve__” は、SX-Aurora TSUBASAでデフォルトで有効になる

```
#ifndef __ve__
#include <xmmintrin.h>
#endif /* __ve__ */
```

# コンパイルエラー(1行が長すぎる)

## 1行が長すぎるコンパイルエラー

- SX-ACEでは固定フォーマットでコンパイル出来るが、
  SX-Aurora TSUBASAではコンパイルエラーとなる場合がある

  コードの例

```
<Tab>   AA = ...空白を含む右辺...*( )        # 行頭がTabコード、最後の ) が 73カラム目
```

  - 最初のタブコードの次の文字が数字以外なら、その文字は7カラム目に現れたとみなされる
  - SX-ACEでは空白を除去してコンパイルするため、カラム数制限内に収まっていた

- 対策
  - 継続文字を使って改行する
  - 自由形式オプションを指定する (-ffree-form)

# ホレリス定数と文字列

## ホレリス定数と文字列でのコンパイルエラー

- DATA文の 初期化記述でコンパイルエラーとなる

```
Error: XXX.f : Data-statement-value incompatible with data-statement-object
```

- 対策
  - 宣言を加える

```
CHARACTER  CNAME*3  # 追加
DATA       CNAME /  'ABC' /
```

- 補足
  - ホレリス定数の代わりとした文字定数の扱い
    SX-ACE　　：記述可能　(コンパイル OK)
    SX-Aurora　TSUBASA ：記述不可　(コンパイル NG)

# サブルーチン引数の間違い

## call文でスカラ変数を配列で受ける

- 状況
  SX-ACE と SX-Aurora TSUBASAで明らかに結果が異なり、
  違う処理を行っていた

  - 元のコードイメージ
    スカラ変数 で call し、呼ばれる側は配列で定義している。
    SX-ACE と SX-Aurora TSUBASA では、呼ばれる側で先頭要素以外で差異があった。

```
LOGICAL    OP
CALL  AA ( , , OP , , )

SUBROUTINE  AA ( , , OPOUT , , )
LOGICAL   OPOUT  ( 2 )          # SX-ACE では 第2要素以降は FALSE で初期化され、
                                # SX-Aurora TSUBASA は不定
IF ( OPOUT(2) ) THEN            # 第2要素の違いで、FLAGの相違になる
  HT =  'TYPE1'
ELSE IF ( OPOUT(1) ) THEN       # 第1要素は引数と一致している
  HT =  'TYPE2'
ENDIF
```

# サブルーチン引数の間違い

## call文でスカラ変数を配列で受ける (続き)

- 修正内容
  - 受ける側の引数をスカラ変数に修正 (callする側を配列にする方法もある)
  - 修正後のコードイメージ

```
LOGICAL    OP
CALL  AA ( , , OP , , )

SUBROUTINE  AA ( , , OPOUT , , )
LOGICAL   OPOUT
LOGICAL   OPOUT_A  ( 2 )

OPOUT_A(1) = OPOUT
OPOUT_A(2) = OPOUT

IF ( OPOUT_A(2) ) THEN
  HT =  'TYPE1'
ELSE IF ( OPOUT_A(1) ) THEN
  HT =  'TYPE2'
ENDIF
```

- 原因
  - 先頭要素以外の値について、初期値に差異がある

| | |
|---|---|
| SX-ACE | 0初期化 あるいは FALSE初期化 |
| SX-Aurora TSUBASA | 不定(LOGICAL では T/F で不定) |

# 演習問題3

別紙「SX-Aurora TSUBASAへの移植について　演習問題」の「演習問題3. 初期化の違い（引数ミス）」を実施します。

# 言語MIXでの構造体受け渡し

## Fortran派生体とC構造体の受け渡し

- 言語MIXで構造体を受け渡すときの注意点
- Fortran側で最適化によりメンバーの宣言順とメモリ割り付けが変わることがある。
  特に文字列が構造体に含まれる場合、文字列が後ろになることが多い(経験則)
  (順序が変わることで想定外の値を渡し、全く関係ない部分でエラー発生となる)

- 対策1
  Fortranの 派生型の定義部分に **"sequence"** 属性を追記する
  (メンバーが定義した順にメモリーに割り付けられる)修正内容

```
type, public :: str_data
  sequence       # 追加
  character(len=FIO_HSHORT) :: vname
  character(len=FIO_HMID)   :: description
  中略
  real(DP)  :: value
 endtype strdata
```

# 言語MIXでの構造体受け渡し

## Fortran派生体とC構造体の受け渡し（続き）

- 対策2
  Fortranの 組込みモジュールISO_C_BINDING を利用する。
  派生型定義でBIND(C)を指定し、Fortranの派生型 と Cの構造型とが同じ個数の成分をもち、Fortranの派生型の成分が、Cの構造型の対応する成分の型と相互利用可能である型及び 型パラメタをもつ場合、Fortranの派生型と言語Cの構造型とが相互利用可能。

  対策2 の 文字型のメンバーは、1バイトの文字型の配列(相互利用可能である型)とする必要がある。

- 補足

  SX-ACE
  -f2003オプションを指定すると、そのサブオプションの”cbind”が、自動的に設定され、
  ”C 相互運用機能を使用することを指定” したことになる

  SX-Aurora TSUBASA
  -std=f2003 オプションを指定しても、cbind特性が指定されないため、
  ソース側で対応が必要となる

# ベクトル化について

## ベクトル化

- SX-ACE ではベクトル化出来ているが、SX-Aurora TSUBASA ではベクトル化出来ていない場合
  - 可能であれば、SX-ACE の編集リスト/最適化メッセージ、ftrace結果と比較し、該当部分の状況を確認する

  - コンパイラ指示行変換ツール で 指示行を変換する
    - Fortran　nfdirconv コマンド
    - C/C++　 ncdirconv コマンド

    詳細は
    Fortran Compiler ユーザーズガイド
    C/C++ Compiler ユーザーズガイド
    の付録(コマンドの使い方、指示行対応表)を参照（詳細はP40）

    変換ツールは単純な置き換えであるため、
    内側ループの展開(指示行 expand) と外側ループのベクトル化で意図しない結果となる場合は、
    指示行 unroll または unroll_completely を試す

  - コンパイラオプション **-fnamed-noalias-aggressive** (Fortran)を試す

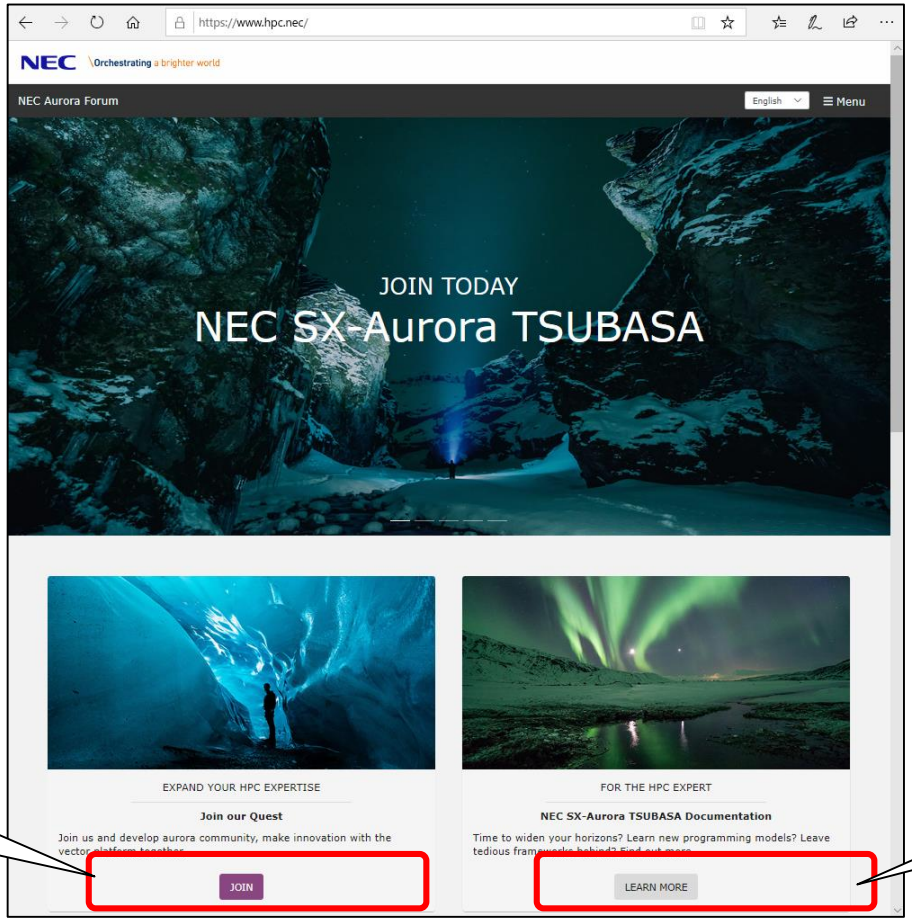
    ポインタの指示先がエイリアスをもたないと仮定して
    より激しく最適化・自動ベクトル化 を適用する。

# 5. マニュアル情報

# マニュアル情報について

## NEC Aurora Forum

- https://www.hpc.nec/
- 各種ドキュメント、コミュニティ




**Aurora Web Forum**

**ドキュメント**

# マニュアル情報について

## 各種ドキュメント

- SDK (コンパイラ情報等)
  - コンパイラオプション、ライブラリ、性能分析ツール等のマニュアル
- NLC（NEC Numeric Library）
- NEC MPI

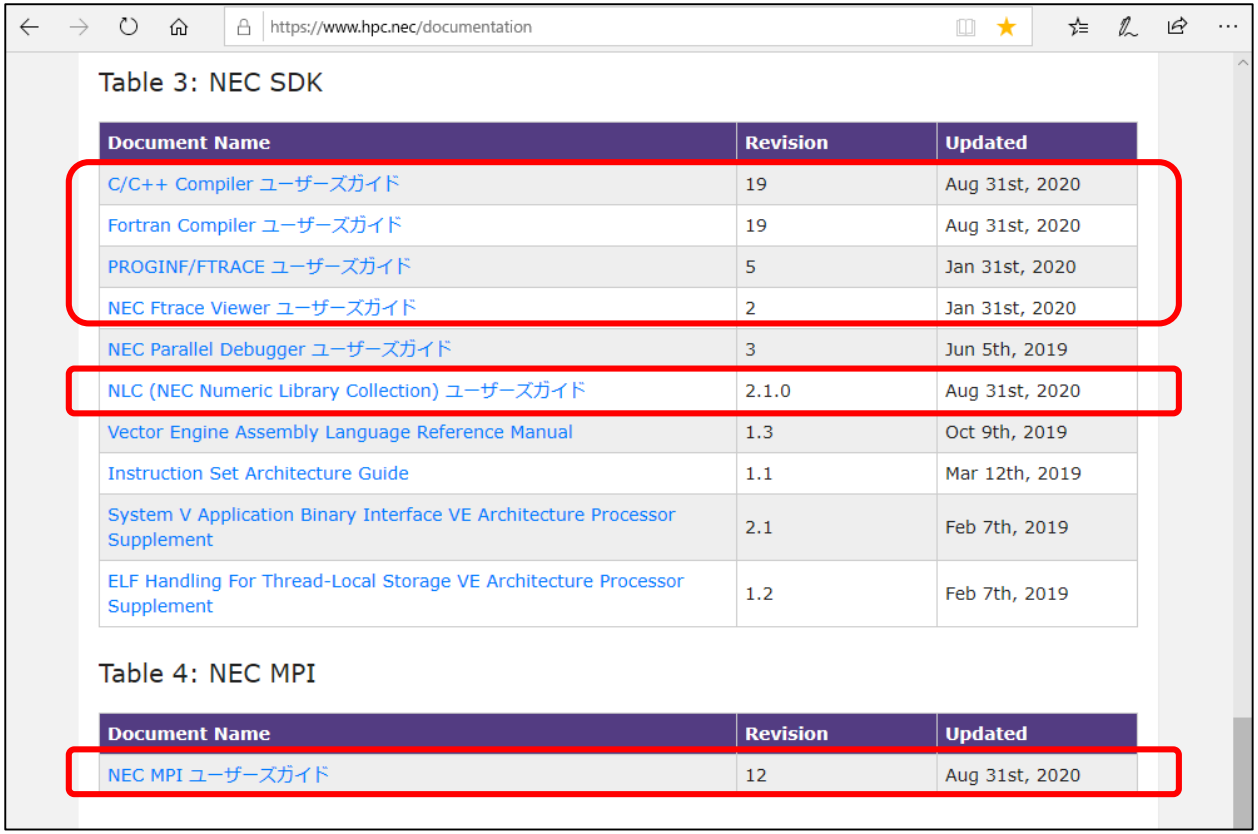



Table 3: NEC SDK

| Document Name | Revision | Updated |
|---|---|---|
| C/C++ Compiler ユーザーズガイド | 19 | Aug 31st, 2020 |
| Fortran Compiler ユーザーズガイド | 19 | Aug 31st, 2020 |
| PROGINF/FTRACE ユーザーズガイド | 5 | Jan 31st, 2020 |
| NEC Ftrace Viewer ユーザーズガイド | 2 | Jan 31st, 2020 |
| NEC Parallel Debugger ユーザーズガイド | 3 | Jun 5th, 2019 |
| NLC (NEC Numeric Library Collection) ユーザーズガイド | 2.1.0 | Aug 31st, 2020 |
| Vector Engine Assembly Language Reference Manual | 1.3 | Oct 9th, 2019 |
| Instruction Set Architecture Guide | 1.1 | Mar 12th, 2019 |
| System V Application Binary Interface VE Architecture Processor Supplement | 2.1 | Feb 7th, 2019 |
| ELF Handling For Thread-Local Storage VE Architecture Processor Supplement | 1.2 | Feb 7th, 2019 |

Table 4: NEC MPI

| Document Name | Revision | Updated |
|---|---|---|
| NEC MPI ユーザーズガイド | 12 | Aug 31st, 2020 |

# マニュアル情報について

## Aurora Web Forums

- 各種情報交換の場として

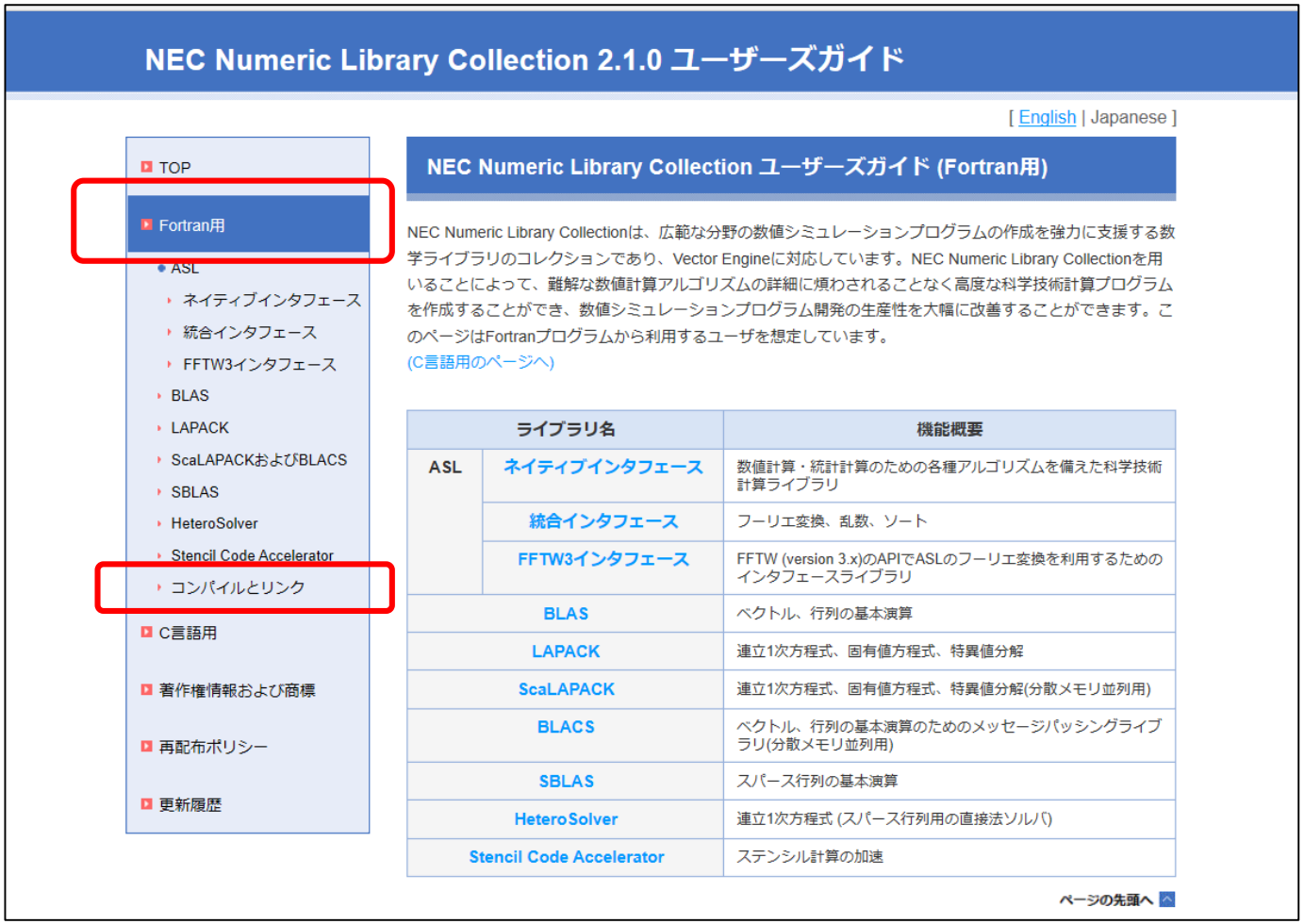
- 項目
  Announce , Primer , Aurora Administration ,
  Software/tools , Tuning , Porting , Open Discussion ,
  NEW BOARDS request , About Aurora Forum

- お役立ち情報の一例
  - Primer
    https://www.hpc.nec/forums/topic?id=p47cxv
    (How to use NEC Compilers for Vector Engine )　ベクトル化、並列化、コンパイラの説明
  - Porting
    https://www.hpc.nec/forums/topic?id=n7NcZy
    (hdf5-1.10.5 (enabling Fortran)　)　HDF5ライブラリの移植方法
  - Tuning
    https://www.hpc.nec/forums/topic?id=p8kc9Z
    (Aurora Vectorization Training )　チューニング方法の具体例(基本から高度まで)
    サンプルコード有

# マニュアル情報について

## NEC Numeric Library Collection マニュアル

- コンパイルとリンク(言語別にある)



NEC Numeric Library Collection 2.1.0 ユーザーズガイド

[ English | Japanese ]

- TOP
- Fortran用
  - ASL
    - ネイティブインタフェース
    - 統合インタフェース
    - FFTW3インタフェース
  - BLAS
  - LAPACK
  - ScaLAPACKおよびBLACS
  - SBLAS
  - HeteroSolver
  - Stencil Code Accelerator
  - コンパイルとリンク
- C言語用
- 著作権情報および商標
- 再配布ポリシー
- 更新履歴

### NEC Numeric Library Collection ユーザーズガイド (Fortran用)

NEC Numeric Library Collectionは、広範な分野の数値シミュレーションプログラムの作成を強力に支援する数学ライブラリのコレクションであり、Vector Engineに対応しています。NEC Numeric Library Collectionを用いることによって、難解な数値計算アルゴリズムの詳細に煩わされることなく高度な科学技術計算プログラムを作成することができ、数値シミュレーションプログラム開発の生産性を大幅に改善することができます。このページはFortranプログラムから利用するユーザを想定しています。

(C言語用のページへ)

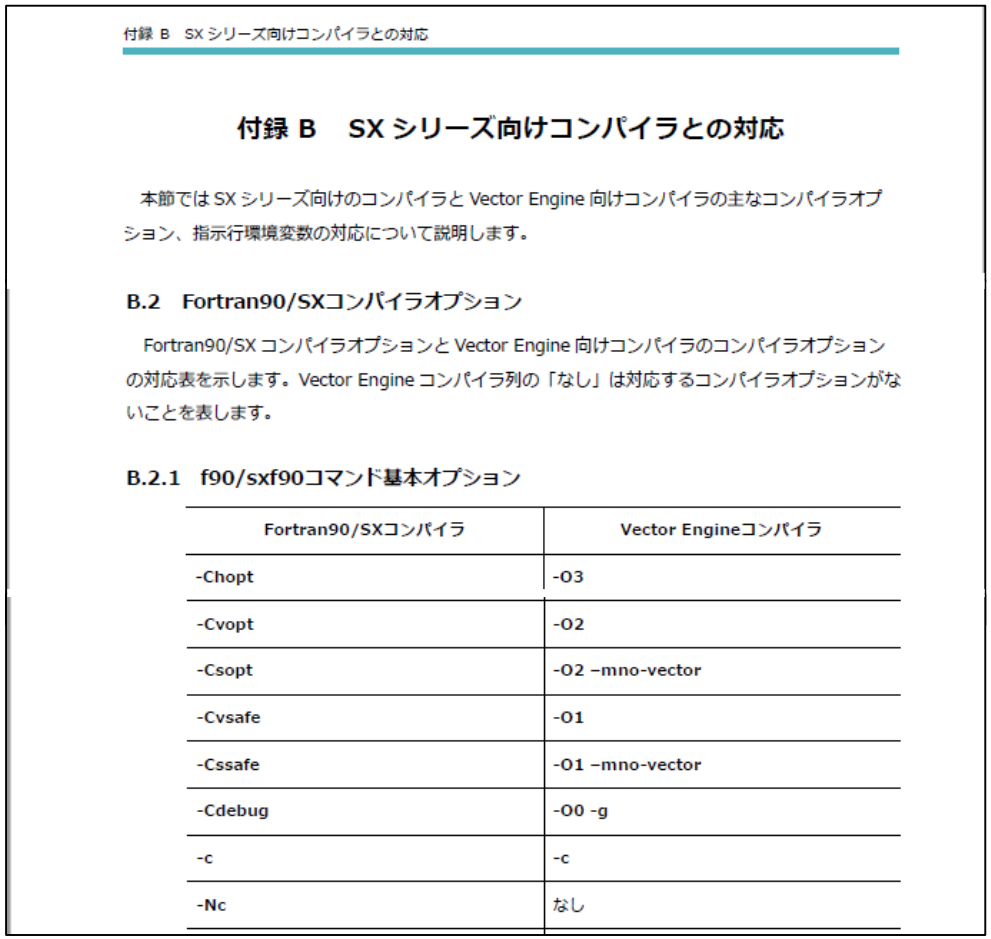
| ライブラリ名 | | 機能概要 |
|---|---|---|
| ASL | ネイティブインタフェース | 数値計算・統計計算のための各種アルゴリズムを備えた科学技術計算ライブラリ |
| | 統合インタフェース | フーリエ変換、乱数、ソート |
| | FFTW3インタフェース | FFTW (version 3.x)のAPIでASLのフーリエ変換を利用するためのインタフェースライブラリ |
| BLAS | | ベクトル、行列の基本演算 |
| LAPACK | | 連立1次方程式、固有値方程式、特異値分解 |
| ScaLAPACK | | 連立1次方程式、固有値方程式、特異値分解(分散メモリ並列用) |
| BLACS | | ベクトル、行列の基本演算のためのメッセージパッシングライブラリ(分散メモリ並列用) |
| SBLAS | | スパース行列の基本演算 |
| HeteroSolver | | 連立1次方程式 (スパース行列用の直接法ソルバ) |
| Stencil Code Accelerator | | ステンシル計算の加速 |

ページの先頭へ

# マニュアル情報について

## コンパイラオプションの対応表 (部分)

- Fortran Compiler ユーザーズガイド 付録B



付録 B　SX シリーズ向けコンパイラとの対応

### 付録 B　SX シリーズ向けコンパイラとの対応

本節では SX シリーズ向けのコンパイラと Vector Engine 向けコンパイラの主なコンパイラオプション、指示行環境変数の対応について説明します。

#### B.2　Fortran90/SXコンパイラオプション

Fortran90/SX コンパイラオプションと Vector Engine 向けコンパイラのコンパイラオプションの対応表を示します。Vector Engine コンパイラ列の「なし」は対応するコンパイラオプションがないことを表します。

#### B.2.1　f90/sxf90コマンド基本オプション

| Fortran90/SXコンパイラ | Vector Engineコンパイラ |
|---|---|
| -Chopt | -O3 |
| -Cvopt | -O2 |
| -Csopt | -O2 –mno-vector |
| -Cvsafe | -O1 |
| -Cssafe | -O1 –mno-vector |
| -Cdebug | -O0 -g |
| -c | -c |
| -Nc | なし |

# マニュアル情報について

## 環境変数の対応表 (部分)

- Fortran Compiler ユーザーズガイド 付録B

**B.4　環境変数**

SX シリーズ向けコンパイラの主要な実行時に参照される環境変数とほぼ同等の機能をもつ Vector Engine 向けコンパイラの環境変数を以下に示します。

| SXシリーズ向けコンパイラ | Vector Engineコンパイラ |
|---|---|
| F_PROGINF | VE_PROGINF |
| F_TRACEBACK | VE_TRACEBACK |
| F_EXPRCW | VE_FORT_EXPRCW |
| F_FMTBUF | VE_FORT_FMTBUF |

## その他ライブラリ (部分)

- Fortran Compiler ユーザーズガイド 付録B

**B.5　その他のライブラリ**

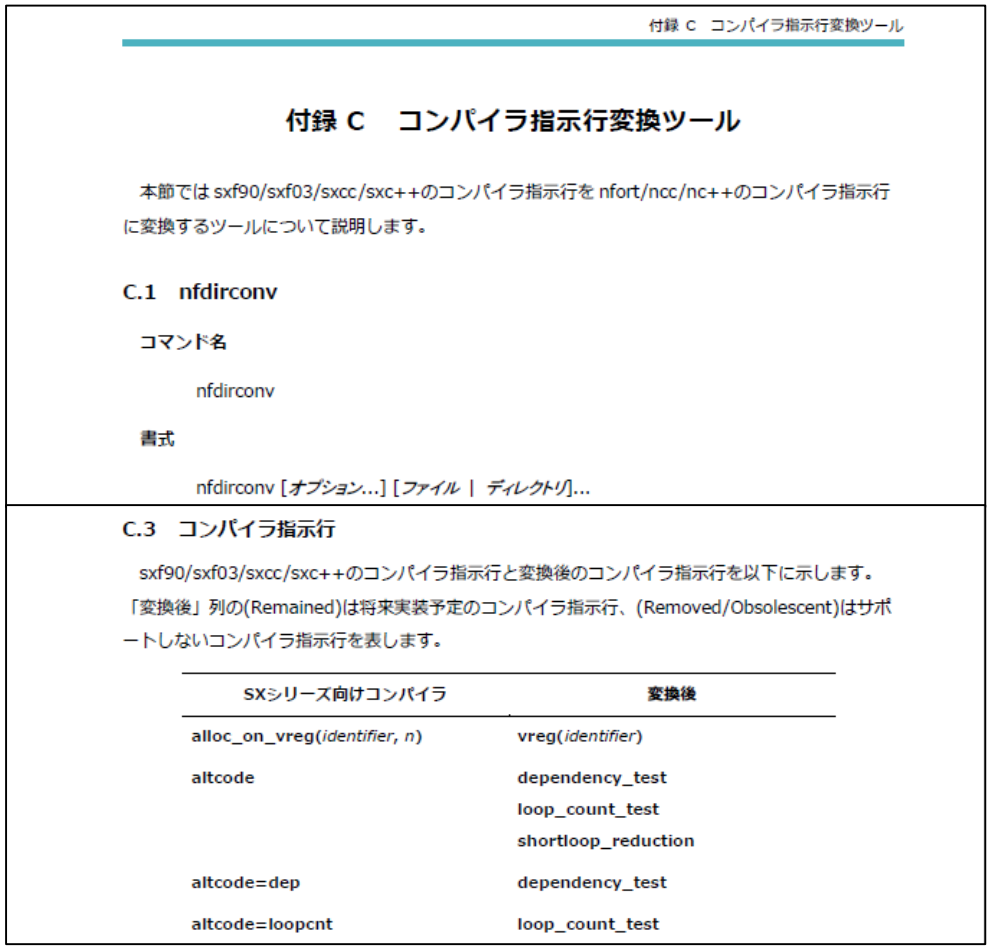
SX シリーズ向けコンパイラのその他のライブラリの手続とほぼ同等の機能をもつ Vector Engine 向けコンパイラの手続を以下に示します。**USE** 文の代わりに**-use** の使用も可能です。

| SXシリーズ向けコンパイラ | Vector Engineコンパイラ |
|---|---|
| CALL ABORT() | USE F90_UNIX<br>CALL ABORT() |
| RESULT = ACCESS(NAME,MODE) | USE F90_UNIX_FILE<br>CALL ACCESS(NAME,AMODE,RESULT)<br>(注) MODE(文字)がAMODE(整数)に変更されています。AMODE(整数)の詳細は「9.3.5 F90_UNIX_FILE」のパラメタを参照してください。 |

# マニュアル情報について

## コンパイラ指示行変換ツールによる対応表 (部分)

- Fortran Compiler ユーザーズガイド 付録C





### 付録 C　コンパイラ指示行変換ツール

本節では sxf90/sxf03/sxcc/sxc++のコンパイラ指示行を nfort/ncc/nc++のコンパイラ指示行に変換するツールについて説明します。

#### C.1　nfdirconv

**コマンド名**

nfdirconv

**書式**

nfdirconv [*オプション...*] [*ファイル* | *ディレクトリ*]...

#### C.3　コンパイラ指示行

sxf90/sxf03/sxcc/sxc++のコンパイラ指示行と変換後のコンパイラ指示行を以下に示します。「変換後」列の(Remained)は将来実装予定のコンパイラ指示行、(Removed/Obsolescent)はサポートしないコンパイラ指示行を表します。

| SXシリーズ向けコンパイラ | 変換後 |
|---|---|
| alloc_on_vreg(*identifier*, *n*) | vreg(*identifier*) |
| altcode | dependency_test<br>loop_count_test<br>shortloop_reduction |
| altcode=dep | dependency_test |
| altcode=loopcnt | loop_count_test |