NEC

SX-Aurora TSUBASA

SX-Aurora TSUBASA
プログラム実行クイックガイド

輸出する際の注意事項

本製品（ソフトウェアを含む）は、外国為替および外国貿易法で規定される規制貨物（または役務）に該当することがあります。

その場合、日本国外へ輸出する場合には日本国政府の輸出許可が必要です。

なお、輸出許可申請手続きにあたり資料等が必要な場合には、お買い上げの販売店またはお近くの当社営業拠点にご相談ください。

は　し　が　き

『SX-Aurora TSUBASA プログラム実行クイックガイド』は、SX-Aurora TSUBASA をはじめてお使いになる方のためのドキュメントです。

SX-Aurora TSUBASA の簡単な使用方法について説明しています。

このガイドでは、SX-Aurora TSUBASA を使用するための VEOS および関連するソフトウェアのインストールが完了していること ならびに ユーザがログインできるようまたはジョブスケジューラ NQSV(NEC Network Queuing System V)もしくは PBS を使用できるよう準備ができていることを前提としています。

またユーザは Fortran コンパイラ(nfort)、C コンパイラ(ncc)、C++コンパイラ(nc++)、NEC MPI の知識があることを前提としています。

このガイドは VEOS のバージョンが 2.3.0 以降であることを前提にしています。VEOS のバージョンは、以下の方法で確認できます。

```
$ rpm -q veos
veos-2.3.0-1.el7.x86_64
```

VH/VE ハイブリッド実行は NEC MPI バージョン 2.3.0 以降で利用できます。NEC MPI のバージョンは以下のディレクトリ/opt/nec/ve/mpi/<version>と対応しています。

```
$ ls -d /opt/nec/ve/mpi/2.3.0
/opt/nec/ve/mpi/2.3.0
```

備考

(1) Linux は Linus Torvalds氏の米国およびその他の国における登録商標あるいは商標です。

(2) InfiniBand は、InfiniBand Trade Association の商標またはサービスマークです。

(3) その他、記載されている会社名、製品名は、各社の登録商標または商標です。

用語定義・略語

用語・略語	説　明
ベクトルエンジン (VE、Vector Engine)	SX-Aurora TSUBASAの中核であり、ベクトル演算を行う部分です。PCI Expressカードであり、x86サーバーに搭載して使用します。
ベクトルホスト (VH、Vector Host)	ベクトルエンジンを保持するサーバー、つまり、ホストコンピュータを指します。
IB	InfiniBandの略語です。
HCA	Host Channel Adapter の略です。 InfiniBand を使用して他ノードと通信するためのハードウェアです。
MPI	Message Passing Interfaceの略語です。主にノード間で並列コンピューティングを行うための標準規格です。同一ノード上のプロセス間の通信にMPIを使用することも可能です。OpenMPとの併用も可能です。
PBS	Altair Engineering社のジョブスケジューラ。商用版のPBS ProfessionalおよびオープンソースのOpenPBSがあります。
chunk	PBSのジョブスクリプト中で指定する要求資源の1単位。1つのchunkで要求された資源は、1台のVHから割り当てられる。
chunk set	1つ以上の同一のchunkの組

目 次

第1章　　SX-Aurora TSUBASA の概要

SX-Aurora TSUBASA は、アプリケーション演算処理を行うベクトルエンジン(VE)と、主に OS 処理を行う x86/Linux ノード(VH)により構成されます。

SX-Aurora TSUBASA のプログラムは、OS 機能を提供する VH から起動し、各 VE 上で実行されます。そのため、SX-Aurora TSUBASA プログラムを実行する際は、VE 番号や VE 数を指定して実行する必要があります。

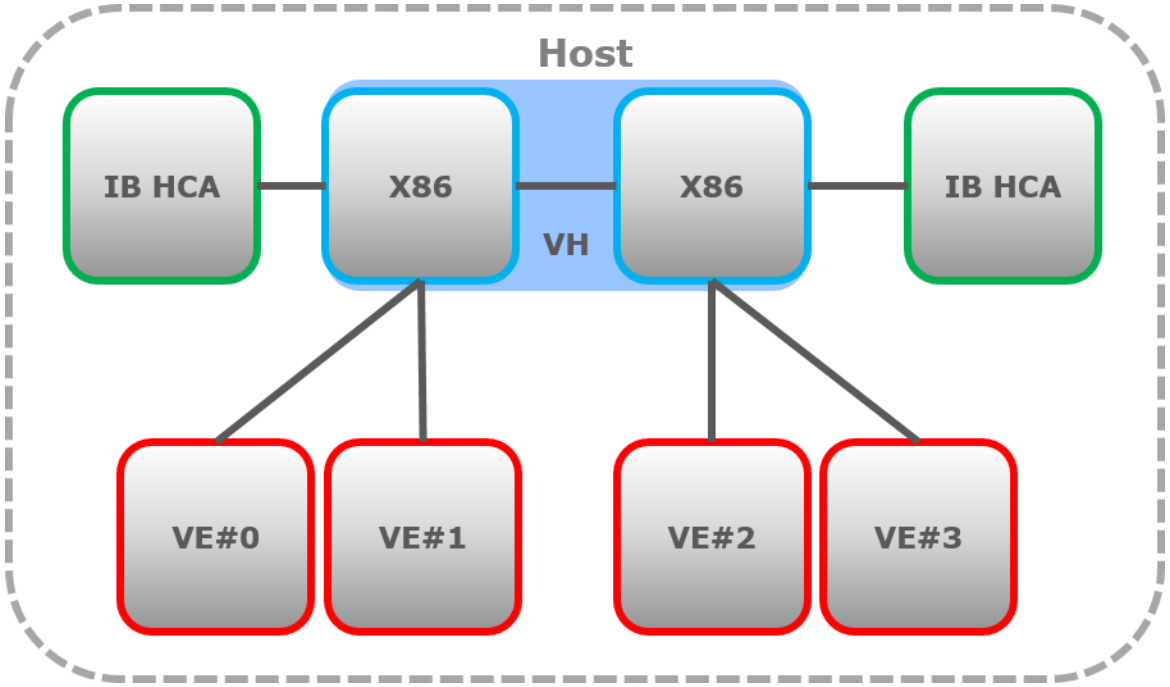

図 1 SX-Aurora TSUBASA の構成例

1.1　VE構成の確認方法

vecmd コマンドにより、VE、HCA(IB) の構成状況を取得することができます。

```
$ /opt/nec/ve/bin/vecmd topo tree
Vector Engine MMM-Command v1.1.2
Command:
topo -N 0,1 tree
--------------------------------------------------------------------------------
SYS-1028GQ-TRT
(QPI Link)
+-80:00.0-+-80:02.0---82:00.0 [VE0] [SOCKET1]
          +-80:03.0---83:00.0 [VE1] [SOCKET1]
          +-80:01.0---81:00.0 [IB0] [SOCKET1] mlx5_0
--------------------------------------------------------------------------------
   Result: Success
```

表示された VE0、VE1 の数字部分が VE 番号です。

第2章　　コンパイル方法

2.1　FORTRAN/C/C++ のコンパイル方法

```
（Fortranの場合）
$ /opt/nec/ve/bin/nfort a.f90
（Cの場合）
$ /opt/nec/ve/bin/ncc    a.c
（C++の場合）
$ /opt/nec/ve/bin/nc++   a.cpp
```

OpenMP を使用する場合、次のようにオプション-fopenmp を指定してください。

```
（Fortranの場合）
$ /opt/nec/ve/bin/nfort -fopenmp a.f90
（Cの場合）
$ /opt/nec/ve/bin/ncc -fopenmp a.c
（C++の場合）
$ /opt/nec/ve/bin/nc++ -fopenmp a.cpp
```

2.2　MPIプログラムのコンパイル方法

NEC MPI を利用する前に、以下のコマンドを実行してください。NEC MPI を利用するために必要な環境変数が設定されます。この設定は、ログアウトするまでは有効ですが、ログインするたびに再実行してください。

```
（bash の場合）
$ source /opt/nec/ve/mpi/<version>/bin/necmpivars.sh
（csh の場合）
% source /opt/nec/ve/mpi/<version>/bin/necmpivars.csh
```

上記の <version> は、ご使用になる NEC MPI のバージョン番号に対応するディレクトリ名です。

MPI プログラムのコンパイルは、次のように、各プログラミング言語に対応した MPI コンパイルコマンドを使用して行います。

```
（Fortranの場合）
$ mpinfort a.f90
（Cの場合）
```

```
$ mpincc   a.c
(C++の場合)
$ mpinc++  a.cpp
```

VH 上で動作する MPI プログラムをコンパイルする場合、-vh オプションを指定します。gfortran, gcc, g++を使用して MPI プログラムがコンパイルされます。

```
(Fortranの場合)
$ mpinfort -vh a.f90
(Cの場合)
$ mpincc   -vh a.c
(C++の場合)
$ mpinc++  -vh a.cpp
```

第3章　　実行方法

3.1　インタラクティブでプログラムを実行する方法

3.1.1　FORTRAN/C/C++ の実行方法

(1)　1 VEモデルの場合

直接プログラムを実行してください。

```
$ ./a.out
```

(2) 1 VEモデル以外で、VE番号を指定してプログラムを実行する方法

ve_exec -N コマンドを使う方法と、環境変数 VE_NODE_NUMBER に指定する方法とがあります。以下の例は、VE#1 を使用する場合です。

- ve_exec -Nを使う方法

```
$ /opt/nec/ve/bin/ve_exec -N 1 a.out
```

- 環境変数に指定する方法

```
(bash の場合)
$ export VE_NODE_NUMBER=1
$ ./a.out
(csh の場合)
% setenv VE_NODE_NUMBER 1
% ./a.out
```

注1) VE番号を指定せずに、
　　$ ./a.out
と実行した場合、a.out は VE#0 で実行されます。

注2) 環境変数VE_NODE_NUMBERおよびコマンドve_exec -NでVE番号を同時に指定した場合、コマンドve_exec -Nの指定が優先されます。

注3) OpenMPスレッドの個数は、環境変数OMP_NUM_THREADSまたはVE_OMP_NUM_THREADSで指定できます。両方が指定された場合、VE上のプログラムに対しては、環境変数VE_OMP_NUM_THREADSの指定が優先されます。

3.1.2　MPI プログラムの実行方法

コンパイル時と同様に、まず次のコマンドを実行してください。この設定は、ログアウトするまでは有効ですが、ログインするたびに再実行してください。

```
(bash の場合)
$ source /opt/nec/ve/mpi/<version>/bin/necmpivars.sh
(csh の場合)
% source /opt/nec/ve/mpi/<version>/bin/necmpivars.csh
```

(1) 一つのVE上でMPIプログラムを実行する方法

mpirun コマンド または mpiexec コマンドに、MPI 実行ファイルを指定して実行します。

この際、-np オプションに、起動する MPI プロセスの個数を指定し、-ve オプションに、使用する VE 番号を指定します。

-np オプションを省略した場合、起動する MPI プロセスの個数は 1 です。

-ve オプションを省略した場合、VE#0 が使用されます。

例えば、VE#3 上で 4 プロセスを使用して MPI プログラムを実行する場合、次のように指定します。

```
$ mpirun -ve 3 -np 4 ./a.out
```

(2)　一つのVH上の複数のVEを使用して実行する方法

-ve オプションで VE 番号の範囲を指定し、-np オプションでプロセスの総数を指定します。

例えば、VE#0 から VE#7 上で合計 16 プロセス(VE あたり 2 プロセスずつ)を使用して実行する場合、次のように指定します。

```
$ mpirun -ve 0-7 -np 16 ./a.out
```

(3)　複数のVH上の複数のVEを使用して実行する方法

-host オプションで VH 名を指定します。

例えば、2 つの VH (host1, host2)を使用して、各 VH の VE#0 および VE#1 上で 16 プロセスずつ

(VE あたり 8 プロセスずつ、計 32 プロセス)を使用して実行する場合、次のように指定します。

```
$ mpirun -host host1 -ve 0-1 -np 16 -host host2 -ve 0-1 -np 16 ./a.out
```

(4)　VHとVEを使用してハイブリッド実行する方法

-vh オプションに続いて VH 上で起動する MPI プロセスの個数、MPI 実行ファイル等を指定します。":"を使用して、VH 上で起動する MPI 実行ファイルと VE 上で起動する MPI 実行ファイルを分けて指定します。

例えば、2 つの VH (host1, host2)を使用して、host1 上で 4 プロセスを使用して MPI 実行ファイル vh.out を実行すると同時に、各 VH の VE#0 および VE#1 上で 16 プロセスずつ(VE あたり 8 プロセスずつ、計 32 プロセス)を使用して MPI 実行ファイル ve.out を連携して実行する場合、次のように指定します。

```
$mpirun -vh -host host1 -np 4 vh.out : -host host1 -ve 0-1 -np 16 -host host2 -ve 0-1 -np 16 ./ve.out
```

3.2　NQSVを利用して、プログラムを実行する場合

ここでは NQSV を利用して、SX-Aurora TSUBASA のプログラムを実行する方法について、説明します。実行方法は一例で、プログラム実行に関係する手続きだけを記述しています。NQSV の詳細については、『NEC Network Queuing System V(NQSV)利用の手引[操作編]』を参照してください。

3.2.1　ジョブの実行方法

ジョブの実行方法にはバッチ実行と会話ジョブがあります。

- バッチ実行
  スクリプトを作成し、qsub コマンドで投入します。
- 会話ジョブ
  qlogin コマンドにより、インタラクティブ実行が可能です。

3.2.2　FORTRAN/C/C++ の実行方法

バッチ実行する場合の FORTRAN/C/C++ スクリプト例です。1VE を使用し、SX-Aurora TSUBASA プログラムを実行します。

```
(script.sh)
:
#PBS --cpunum-lhost=1    # Number of CPUs
#PBS --venum-lhost=1     # Number of VE
./a.out
```

以下のように qsub コマンドで投入します。

```
$ /opt/nec/nqsv/bin/qsub script.sh
```

会話ジョブの場合も同様に

```
$ /opt/nec/nqsv/bin/qlogin --venum-lhost=1 …
$ ./a.out
```

として実行してください。

注) VE の割り当てはNQSVが行います。
ユーザは、環境変数 VE_NODE_NUMBER と ve_exec -N を指定しないでください。

3.2.3　MPI プログラムの実行方法

(1) NQSVに割り当てられたVEの内、特定のVEを使用する方法

バッチ実行において、例えば、論理ホスト#0の4つのVE上で、合計32プロセス(各VE上で8プロセスずつ)を使用して実行する場合、次のように指定します。

```
(script2.sh)
:
#PBS --cpunum-lhost=1    # Number of CPUs
#PBS --venum-lhost=4     # Number of VEs
source /opt/nec/ve/mpi/<version>/bin/necmpivars.sh
mpirun -host 0 -ve 0-3 -np 32 ./a.out
```

以下のように qsub コマンドで投入します。

```
$ /opt/nec/nqsv/bin/qsub script2.sh
```

(2)　NQSVに割り当てられた全てのVEを使用して実行する方法

例えば、4 つの論理ホストを使用して、各論理ホストの 8 つの VE 上で合計 32 プロセス(各 VE 上で 1 プロセスずつ)を使用して実行する場合、次のように指定します。

```
 (script3.sh)
:
#PBS -T necmpi
#PBS -b 4               # Number of logical hosts
#PBS --cpunum-lhost=1   # Number of CPUs
#PBS --venum-lhost=8    # Number of VEs per logical host
#PBS --use-hca=2        # Number of available HCAs
source /opt/nec/ve/mpi/<version>/bin/necmpivars.sh
mpirun -np 32 ./a.out
```

以下のように qsub コマンドで投入します。

```
$ /opt/nec/nqsv/bin/qsub script3.sh
```

(3)　NQSVに割り当てられた全てのVHとVEを使用してVH-VEハイブリッドMPI実行する方法

例えば、4 つの論理ホストを使用して、各論理ホスト上で合計 12 プロセス(各 VH 上で 3 プロセスずつ)を使用して MPI 実行ファイル vh.out を実行すると同時に、各論理ホストの 8 つの VE 上で合計 32 プロセス(各 VE 上で 1 プロセスずつ)を使用して MPI 実行ファイル ve.out を連携して実行する場合、次のように指定します。

```
 (script4.sh)
:
#PBS -T necmpi
#PBS -b 4               # Number of logical hosts
#PBS --cpunum-lhost=4   # Number of CPUs per logical host
#PBS --venum-lhost=8    # Number of VEs per logical host
#PBS --use-hca=2        # Number of available HCAs
source /opt/nec/ve/mpi/<version>/bin/necmpivars.sh
mpirun -vh -np 12 vh.out : -np 32 ./ve.out
```

以下のように qsub コマンドで投入します。

```
$ /opt/nec/nqsv/bin/qsub script4.sh
```

会話ジョブの場合も実行方法は同じです。

注) MPI プロセスの VH や VE への割り当ては、NQSV が自動的に行います。
　ユーザが、実行する VH 名または VE 番号などを指定しないでください。

3.3　PBSを利用してプログラムを実行する場合

本節では、SX-Aurora TSUBASA のプログラムを PBS 配下で実行する方法について説明します。本節の説明は、システムにインストールされた PBS に SX-Aurora TSUBASA 用の設定が完了していることを前提としています。PBS の設定方法の詳細は、「Altair PBS Professional Administrator's Guide」中の「Support for NEC SX-Aurora TSUBASA」の章を参照してください。本節では最も基本的な実行方法を説明します。より高度な実行方法については、「Altair PBS Professional User's Guide」の「Submitting Jobs to NEC SX-Aurora TSUBASA」の章を参照してください。

3.3.1　概要

PBS 配下でバッチジョブを投入する場合，コマンド qsub にジョブスクリプト・ファイルを指定して実行します。対話的にジョブを実行する場合は、コマンド qsub -I にジョブスクリプト・ファイルを指定して実行します。ジョブの状況は，コマンド qstat で確認できます。ジョブを削除する場合、コマンド qdel に削除するジョブのジョブ ID を指定して実行します。これらのコマンドは、通常 /opt/pbs/bin 配下にありますので、コマンドの実行パスを適切に設定してください。

PBS のジョブスクリプト中では、接頭辞"#PBS "で始まる PBS 指示行に、必要な計算資源などを指定します。例えば、使用する VE の台数は資源 nves で指定し、実行する MPI プロセスの個数は資源 mpiprocs で指定します。次の例では、4 台の VE 上で 8 個の MPI プロセスを実行することを指定しています。PBS 指示行の"-l select="で始まる指定を selection 指示行と呼びます。

```
#PBS -l select=nves=4:mpiprocs=8
```

"資源=値"を":"で連結した"nves=4:mpiprocs=8"のような要求資源の単位を chunk と呼びます。1 つの chunk で要求した資源は、必ず 1 つの VH から割り当てられます。したがって、例えば 4 つの VE が接続されている VH に対しては、nves の値が 4 以下になるように指定してください。

同一の chunk を複数個要求する場合、次のように chunk の前に"個数:"を指定します。この場合、1 台の VE 上で 2 個の MPI プロセスを実行する chunk を 4 組要求していることになります。このよ

うな同一の chunk の組を chunk set と呼びます。

```
#PBS -l select=4:nves=1:mpiprocs=2
```

3.3.2　Fortran、C または C++のプログラムの実行方法

次のジョブスクリプトは、1 台の VE を使用して、Fortran、C または C++の SX-Aurora TSUBASA プログラムを実行する例です。

```
#!/bin/bash
#PBS -l select=nves=1

./a.out
```

注) VE の割当てはPBSが自動的に行います。環境変数VE_NODE_NUMBERまたはve_execコマンドのオプション-Nを指定しないでください。

OpenMP プログラムの場合、資源 ompthreads を指定して、スレッドの個数を明示してください。

次のジョブスクリプトは、1 台の VE を使用して 8 個のスレッドからなる OpenMP プログラムを実行する例です。

```
#!/bin/bash
#PBS -l select=nves=1:ompthreads=8

./a.out
```

注) 環境変数OMP_NUM_THREADSの値は、PBSが自動的に設定するので指定しないでください。指定しても、PBSが上書きします。

3.3.3　MPI プログラムの実行方法

(1) VE上でMPIプログラムを実行する場合

次のジョブスクリプトは、各 VE 上で 8 個の MPI プロセスを実行し、4 台の VE で合計 32 個の MPI プロセスを実行する例です。1 台の VE 上で実行する MPI プロセスの個数を指定した chunk ”nves=1:mpiprocs=8”を記述し、その直前に”4:”を付加して使用する VE の台数を指定しています。

mpirun コマンドにはオプション-np を指定し、MPI プロセスの総数を指定してください。

```
#!/bin/bash
#PBS -l select=4:nves=1:mpiprocs=8

source /opt/nec/ve/mpi/<version>/bin/necmpivars.sh
mpirun -np 32 ./a.out
```

(2) OpenMPを併用したハイブリッド並列プログラムを実行する場合

次のジョブスクリプトは、4 個のスレッドからなる MPI プロセスを、各 VE 上で 2 個実行し、8 台の VE を使用して、合計 16 個の MPI プロセスで実行する例です。

```
#!/bin/bash
#PBS -l select=8:nves=1:mpiprocs=2:ompthreads=4

source /opt/nec/ve/mpi/<version>/bin/necmpivars.sh
mpirun -np 16 ./a.out
```

(3) VH-VEハイブリッドMPIプログラムを実行する場合

MPI プロセスを VH 上でも実行する場合、環境変数 NEC_PROCESS_DIST を指定して、各 chunk で実行する MPI プロセスの配置を指定する必要があります。

VH 上で 2 個の MPI プロセスを実行すると同時に、各 VE 上で 4 個の MPI プロセスを実行し、8 台の VE を使用して、合計 34 個の MPI プロセスで VH-VE ハイブリッド MPI プログラムを実行する場合、次のように指定できます。

```
#!/bin/bash
#PBS -l select=ncpus=2:mpiprocs=2+8:nves=1:mpiprocs=4
#PBS -v NEC_PROCESS_DIST=s2+4

source /opt/nec/ve/mpi/<version>/bin/necmpivars.sh
mpirun -vh -np 2 vh.out : -np 32 ./ve.out
```

VH 上で実行する MPI プロセスの場合、PBS 指示行には、上記の”ncpus=2:mpiprocs=2”のように、

MPI プロセスを実行する VH の CPU コアの個数を資源 ncpus で指定してください。この例の”ncpus=2:mpiprocs=2”および”8:nves=1:mpiprocs=4”のように、異なる種類の chunk set を指定する場合、それらを”+”で連結します。

環境変数 NEC_PROCESS_DIST には、selection 指示行の各 chunk set で実行する MPI プロセスの配置を”+”で連結して指定します。上の例では、最初の chunk set は、VH 上で実行する MPI プロセスの個数を文字“s”に続けて指定しています。2 番目の chunk set は、各 VE 上で実行する MPI プロセスの個数を指定しています。

selection 指示行および環境変数 NEC_PROCESS_DIST における VH プロセスおよび VE プロセスの順序は、コマンド mpirun における順序と一致するように指定してください。これは、MPI プロセスのランクが、selection 指示行および環境変数 NEC_PROCESS_DIST で指定した chunk の順番で決定されるためです。

第4章　　I/O 高速化

本章で説明されている環境変数を指定してプログラムを実行することにより、I/O を高速化できます。

4.1　ScaTeFS ダイレクト I/O

read/writeのIOサイズが規定値（デフォルトは 1MB）以上の時に、ScaTeFS に対しダイレクト I/O を行います。実行時に環境変数VE_LD_PRELOADでライブラリ libscatefsib を指定してください。

利用条件：　ScaTeFS をインストールしており、ScaTeFS I/O クライアントが VH に設定済みであること

```
(bash の場合)
$ export VE_LD_PRELOAD=libscatefsib.so.1
$ ./a.out
 (csh の場合)
% setenv VE_LD_PRELOAD libscatefsib.so.1
% ./a.out
```

NQSV を利用してプログラムを実行する場合は、--use-hca に値を設定してください。

```
#!/bin/bash
#PBS -b 1
#PBS --venum-lhost=1
#PBS --use-hca=2        # Number of available HCAs

VE_LD_PRELOAD=libscatefsib.so.1 ./a.out
```

PBS を利用してプログラムを実行する場合、ジョブスクリプト中で環境変数 VE_LD_PRELOAD を設定してください。

```
#!/bin/bash
#PBS -l select=nves=1

VE_LD_PRELOAD=libscatefsib.so.1 ./a.out
```

4.2　高速 I/O

高速I/Oライブラリが、VHとVE間のデータ転送を効率的に行うことで、I/O性能を向上します。実行時に環境変数VE_ACC_IOに1を指定してください。

利用条件：　システム管理者が、「SX-Aurora TSUBASA インストールガイド」の指示に従い、カーネルパラメータ"vm.nr_hugepages"を使用して、高速I/O用にHuge Pagesを予約していること。

```
 (bash の場合)
$ export VE_ACC_IO=1
$ ./a.out
 (csh の場合)
% setenv VE_ACC_IO 1
% ./a.out
```

NQSV を利用してプログラムを実行する場合は、バッチ実行のためのスクリプトの中で環境変数を設定してください。

```
#!/bin/bash
#PBS -b 1
#PBS --venum-lhost=1

export VE_ACC_IO=1
./a.out
```

PBS を利用してプログラムを実行する場合は、同様にジョブスクリプト中で環境変数を設定してください。

```
#!/bin/bash
```

```
#PBS -l select=nves=1


export VE_ACC_IO=1
./a.out
```

なお、環境変数 VE_ACC_IO は VEOS のバージョンが 2.3.0 以降の場合に指定可能です。
VEOS のバージョンは、以下の方法で確認できます。

```
$ rpm -q veos
veos-2.3.0-1.el7.x86_64
```

VEOS のバージョンは 2.3.0 より前の場合は、環境変数 VE_ACC_IO は使えません。その場合は、環境変数 VE_LD_PRELOAD に libveaccio.so.1 を指定してください。

第5章　　プログラムの実行性能を確認する方法

プログラムの実行性能を確認する場合、PROGINF 機能と FTRACE 機能を使用します。

5.1　PROGINF機能

PROGINF はプログラム全体の性能情報を出力する機能です。PROGINF 機能を使用する場合、環境変数 VE_PROGINF に YES または DETAIL を指定してプログラムを実行します。プログラムの実行終了時、プログラム全体の性能情報が出力されます。

```
$ /opt/nec/ve/bin/ncc source.c
$ export VE_PROGINF=YES
$ ./a.out

          ********  Program  Information  ********
  Real Time (sec)                          :           100.795725
  User Time (sec)                          :           100.686826
  Vector Time (sec)                        :            41.125491
  Inst. Count                              :         82751792519
  V. Inst. Count                           :         11633744762
  V. Element Count                         :        881280485102
  V. Load Element Count                    :        268261733727
  FLOP count                               :        625104742151
  MOPS                                     :         11778.920848
  MOPS (Real)                              :         11765.127159
  MFLOPS                                   :          6209.015275
  MFLOPS (Real)                            :          6201.744217
  A. V. Length                             :            75.752090
  V. Op. Ratio (%)                         :            94.002859
  L1 Cache Miss (sec)                      :             6.364831
  VLD LLC Hit Element Ratio (%)            :            90.032527
  Memory Size Used (MB)                    :           918.000000
  Non Swappable Memory Size Used (MB)      :            84.000000

  Start Time (date)         :        Tue Nov 17 12:43:08 2020 JST
  End   Time (date)         :        Tue Nov 17 12:44:49 2020 JST
```

MPI プログラムの場合、環境変数 NMPI_PROGINF に YES または DETAIL を指定してプログラムを実行します。その結果、MPI プログラム実行全体の性能情報が出力されます。

```
$ source /opt/nec/ve/mpi/<version>/bin/necmpivars.sh
$ mpincc source.c
$ export NMPI_PROGINF=YES
$ mpirun -np 4 -ve 0-1 ./a.out

MPI Program Information:
========================
Note: It is measured from MPI_Init till MPI_Finalize.
      [U,R] specifies the Universe and the Process Rank in the Universe.
      Times are given in seconds.


Global Data of 4 Vector processes       :          Min [U,R]          Max [U,R]      Average
================================

Real Time (sec)                         :       258.752 [0,1]      258.769 [0,0]      258.760
User Time (sec)                         :       258.632 [0,0]      258.672 [0,3]      258.661
Vector Time (sec)                       :       163.308 [0,3]      165.063 [0,2]      164.282
Inst. Count                             : 255247993643 [0,0] 255529897274 [0,3] 255372547702
V. Inst. Count                          :  19183106540 [0,0]  19190366299 [0,3]  19186786385
V. Element Count                        : 731572775534 [0,2] 731612551928 [0,3] 731597913441
V. Load Element Count                   : 213554974007 [0,0] 213586395765 [0,3] 213566855461
FLOP Count                              : 580774521087 [0,3] 580807048542 [0,0] 580790784573
MOPS                                    :      4464.705 [0,2]     4465.784 [0,3]     4465.280
MOPS (Real)                             :      4462.927 [0,0]     4464.222 [0,3]     4463.583
MFLOPS                                  :      2245.220 [0,3]     2245.688 [0,0]     2245.373
MFLOPS (Real)                           :      2244.435 [0,3]     2244.588 [0,1]     2244.519
A. V. Length                            :        38.124 [0,3]       38.138 [0,0]       38.130
V. Op. Ratio (%)                        :        79.541 [0,3]       79.559 [0,0]       79.551
L1 Cache Miss (sec)                     :        36.603 [0,2]       38.208 [0,3]       37.331
VLD LLC Hit Element Ratio (%)           :        87.174 [0,1]       87.176 [0,2]       87.175
Memory Size Used (MB)                   :       677.000 [0,1]      933.000 [0,0]      741.000
Non Swappable Memory Size Used (MB)     :       115.000 [0,0]      179.000 [0,2]      131.000


Overall Data of 4 Vector processes
==================================

Real Time (sec)                         :       258.769
User Time (sec)                         :      1034.645
```

```
Vector Time (sec)                          :       657.127
GOPS                                       :        14.966
GOPS (Real)                                :        14.960
GFLOPS                                     :         8.981
GFLOPS (Real)                              :         8.978
Memory Size Used (GB)                      :         2.895
Non Swappable Memory Size Used (GB)        :         0.512


VE Card Data of 2 VEs
=====================

Memory Size Used (MB) Min                  :      1354.000 [node=0,ve=1]
Memory Size Used (MB) Max                  :      1610.000 [node=0,ve=0]
Memory Size Used (MB) Avg                  :      1482.000
Non Swappable Memory Size Used (MB) Min    :       230.000 [node=0,ve=1]
Non Swappable Memory Size Used (MB) Max    :       294.000 [node=0,ve=0]
Non Swappable Memory Size Used (MB) Avg    :       262.000
```

5.2　FTRACE機能

FTRACE は関数ごとの性能情報を測定・出力する機能です。FTRACE 機能を使用する場合、-ftrace オプションを指定してプログラムをコンパイルし、実行します。プログラムの実行終了後、解析情報ファイル(ftrace.out)が出力されます。性能情報を確認するには、解析情報ファイル(ftrace.out)を指定して ftrace コマンドを実行します。

```
$ /opt/nec/ve/bin/nfort -ftrace source.f90
$ ./a.out
$ /opt/nec/ve/bin/ftrace -f ftrace.out

*----------------------*
  FTRACE ANALYSIS LIST
*----------------------*

Execution Date : Tue May  8 15:22:15 2018 JST
Total CPU Time : 0:03'21"561 (201.561 sec.)

FREQUENCY  EXCLUSIVE       AVER.TIME     MOPS   MFLOPS  V.OP  AVER.    VECTOR L1CACHE CPU PORT VLD LLC PROC.NAME
           TIME[sec](  %  )    [msec]                    RATIO V.LEN      TIME    MISS     CONF HIT E.%
```

```
     25100    96.105( 47.7)     3.829    1455.0     728.7  39.20   8.0     46.967   17.785    0.314    93.16 funcA
     25100    82.091( 40.7)     3.271    1703.3     853.1  36.95   7.6     46.462   18.024    0.314    98.29 funcB
  13124848     7.032(  3.5)     0.001     772.7     229.6   0.00   0.0      0.000    4.184    0.000     0.00 funcC
       253     6.007(  3.0)    23.745   35379.0   19138.0  97.21  99.8      5.568    0.181    1.128    89.40 funcD
     25100     3.684(  1.8)     0.147   45327.6   21673.3  98.35 114.3      3.455    0.218    1.076    94.75 funcE
     25100     3.611(  1.8)     0.144   51034.2   25382.3  98.37 111.0      3.451    0.143    1.076    88.64 funcF
         2     2.447(  1.2)  1223.578    1262.9      79.3   0.00   0.0      0.000    1.044    0.000     0.00 funcG
         2     0.317(  0.2)   158.395   32624.9   11884.9  96.79  99.1      0.272    0.034    0.000     7.07 funcH
         1     0.217(  0.1)   216.946    1318.8      69.1   0.00   0.0      0.000    0.089    0.000     0.00 funcI
         2     0.025(  0.0)    12.516    1254.8       0.0   0.00   0.0      0.000    0.011    0.000     0.00 funcJ
         1     0.019(  0.0)    19.367   54199.2   33675.0  97.87 100.3      0.019    0.000    0.010    94.02 funcK
         4     0.004(  0.0)     0.948   57592.4   24101.4  97.88 121.4      0.004    0.000    0.000     4.72 funcL
         1     0.001(  0.0)     0.861     517.9       3.2   0.00   0.0      0.000    0.000    0.000     0.00 funcM
-------------------------------------------------------------------------------------------------------------------
  13225514   201.561(100.0)     0.015    4286.1    2147.5  76.91  34.7    106.197   41.712    3.917    89.99 total
```

MPI の場合、VE 上で動作する MPI プログラムで FTRACE 機能が利用可能です。FTRACE 機能を使用する場合、-ftrace オプションを指定して VE 上で動作する MPI プログラムをコンパイルし、実行します。プログラムの実行終了後、性能情報が MPI プロセスごとに別の解析情報ファイル(*1)に出力されます。

(*1) ファイル名は、「ftrace.out.グループ ID.ランク番号」となります。グループ ID、ランク番号はそれぞれ NEC MPI の環境変数 MPIUNIVERSE、MPIRANK の値です。

```
$ source /opt/nec/ve/mpi/<version>/bin/necmpivars.sh
$ mpinfort -ftrace source.f90
$ mpirun -np 4 ./a.out
$ ls ftrace.out.*
ftrace.out.0.0 ftrace.out.0.1 ftrace.out.0.2 ftrace.out.0.3
$ /opt/nec/ve/bin/ftrace -f ftrace.out.* (MPIプログラム実行全体の測定結果を出力)

*---------------------*
  FTRACE ANALYSIS LIST
*---------------------*

Execution Date : Sat Feb 17 12:44:49 2018 JST
Total CPU Time : 0:03'24"569 (204.569 sec.)
```

```
FREQUENCY  EXCLUSIVE       AVER.TIME     MOPS    MFLOPS  V.OP  AVER.    VECTOR L1CACHE CPU PORT VLD LLC PROC.NAME
           TIME[sec](  % )    [msec]                     RATIO V.LEN      TIME    MISS     CONF HIT E.%

     1012   49.093( 24.0)     48.511  23317.2  14001.4  96.97  83.2    42.132   5.511    0.000   80.32 funcA
   160640   37.475( 18.3)      0.233  17874.6   9985.9  95.22  52.2    34.223   1.973    2.166   96.84 funcB
   160640   30.515( 14.9)      0.190  22141.8  12263.7  95.50  52.8    29.272   0.191    2.544   93.23 funcC
   160640   23.434( 11.5)      0.146  44919.9  22923.2  97.75  98.5    21.869   0.741    4.590   97.82 funcD
   160640   22.462( 11.0)      0.140  42924.5  21989.6  97.73  99.4    20.951   1.212    4.590   96.91 funcE
 53562928   15.371(  7.5)      0.000   1819.0    742.2   0.00   0.0     0.000   1.253    0.000    0.00 funcG
        8   14.266(  7.0)   1783.201   1077.3     55.7   0.00   0.0     0.000   4.480    0.000    0.00 funcH
   642560    5.641(  2.8)      0.009    487.7      0.2  46.45  35.1     1.833   1.609    0.007   91.68 funcF
     2032    2.477(  1.2)      1.219    667.1      0.0  89.97  28.5     2.218   0.041    0.015   70.42 funcI
        8    1.971(  1.0)    246.398  21586.7   7823.4  96.21  79.6     1.650   0.271    0.000    2.58 funcJ
-------------------------------------------------------------------------------------------------------------------
 54851346  204.569(100.0)      0.004  22508.5  12210.7  95.64  76.5   154.524  17.740   13.916   90.29 total

   ELAPSED      COMM.TIME  COMM.TIME   IDLE TIME  IDLE TIME  AVER.LEN      COUNT  TOTAL LEN PROC.NAME
   TIME[sec]        [sec]  / ELAPSED        [sec]  / ELAPSED    [byte]               [byte]

      12.444        0.000                   0.000                 0.0          0        0.0  funcA
       9.420        0.000                   0.000                 0.0          0        0.0  funcB
       7.946        0.000                   0.000                 0.0          0        0.0  funcG
       7.688        0.000                   0.000                 0.0          0        0.0  funcC
       7.372        0.000                   0.000                 0.0          0        0.0  funcH
       5.897        0.000                   0.000                 0.0          0        0.0  funcD
       5.653        0.000                   0.000                 0.0          0        0.0  funcE
       1.699        1.475                   0.756                3.1K     642560        1.9G funcF
       1.073        1.054                   0.987                1.0M       4064        4.0G funcI
       0.704        0.045                   0.045                80.0          4      320.0  funcK
-------------------------------------------------------------------------------------------------------------------

FREQUENCY  EXCLUSIVE       AVER.TIME     MOPS    MFLOPS  V.OP  AVER.    VECTOR L1CACHE CPU PORT VLD LLC PROC.NAME
           TIME[sec](  % )    [msec]                     RATIO V.LEN      TIME    MISS     CONF HIT E.%

     1012   49.093( 24.0)     48.511  23317.2  14001.4  96.97  83.2    42.132   5.511    0.000   80.32 funcA
      253   12.089            47.784  23666.9  14215.9  97.00  83.2    10.431   1.352    0.000   79.40  0.0
      253   12.442            49.177  23009.2  13811.8  96.93  83.2    10.617   1.406    0.000   81.26  0.1
      253   12.118            47.899  23607.4  14180.5  97.00  83.2    10.463   1.349    0.000   79.36  0.2
      253   12.444            49.185  23002.8  13808.2  96.93  83.2    10.622   1.404    0.000   81.26  0.3
:
```

```
------------------------------------------------------------------------------------------------------------
 54851346   204.569(100.0)     0.004  22508.5  12210.7  95.64  76.5   154.524  17.740   13.916   90.29 total

   ELAPSED     COMM.TIME  COMM.TIME   IDLE TIME  IDLE TIME  AVER.LEN      COUNT  TOTAL LEN PROC.NAME
   TIME[sec]       [sec]  / ELAPSED       [sec]  / ELAPSED    [byte]               [byte]

      12.444       0.000                 0.000                 0.0           0       0.0  funcA
      12.090       0.000      0.000      0.000      0.000      0.0           0       0.0   0.0
      12.442       0.000      0.000      0.000      0.000      0.0           0       0.0   0.1
      12.119       0.000      0.000      0.000      0.000      0.0           0       0.0   0.2
      12.444       0.000      0.000      0.000      0.000      0.0           0       0.0   0.3
:
```

5.3　プロファイラの使用方法

コンパイル、リンク時に-pg を指定し、コンパイルしたプログラムを実行すると性能測定結果がファイル gmon.out に出力されます。gmon.out を ngprof コマンドで解析、表示できます。

```
$ /opt/nec/ve/bin/nfort -pg a.f90
$ ./a.out
$ /opt/nec/ve/bin/ngprof ./a.out
(性能情報が表示される)
```

MPI でプロファイラを利用する場合、各プロセスがファイル gmon.out を上書きし合うのを避けるため、VE_GMON_OUT_PREFIX と GMON_OUT_PREFIX 環境変数を使用してプロセスごとに異なるファイル名を指定します。VE 上で動作するプログラムが出力する gmon.out ファイルの名前を変更するには VE_GMON_OUT_PREFIX 環境変数を指定します。VH 上で動作するプログラムが出力する gmon.out ファイルの名前を変更するには GMON_OUT_PREFIX 環境変数を指定します。

以下のラッパースクリプト(gprof-mpi.sh)を経由して実行すると、各 MPI プロセスの性能測定結果が「gmon.out.<MPI ユニバース番号>:<MPI ランク番号>.<pid>」というプロセスごとに異なるファイル名で出力できます。

```
 (gprof-mpi.sh)
#!/bin/bash
# 性能測定結果ファイル名をgmon.out.<MPIユニバース番号>:<MPIランク番号>.<pid>」に変更
export VE_GMON_OUT_PREFIX=gmon.out.${MPIUNIVERSE}:${MPIRANK}
export GMON_OUT_PREFIX=gmon.out.${MPIUNIVERSE}:${MPIRANK}
exec $*
```

```
(MPIの環境を設定)
$ source /opt/nec/ve/mpi/<version>/bin/necmpivars.sh

(MPIプログラムのコンパイル)
$ mpincc -pg a.c -o ve.out
$ mpincc -vh -pg a.c -o vh.out

(gprof-mpi.shスクリプト経由で実行)
$ mpirun -np 1 ./gprof-mpi.sh ./ve.out : -vh -np 1 ./gprof-mpi.sh ./vh.out

$ ls gmon.out.*
gmon.out.0:0.19390  gmon.out.0:1.19391

(ランク0(VE上で動作するMPIプログラム)の結果を表示)
$ /opt/nec/ve/bin/ngprof ve.out gmon.out.0:0.19390

(ランク1(VH上で動作するMPIプログラム)の結果を表示)
$ /usr/bin/gprof vh.out gmon.out.0:1.19391
```

第6章　　一般的な問題と解決策

- **問題**

  Linux でよく知られているコマンドの SX-Aurora TSUBASA 版はありますか。

- **解決**

  以下のような SX-Aurora TSUBASA 用コマンドを提供しています。

  ps, pmap, time, gdb, automake, top, free, vmstat
  これらのコマンドは /opt/nec/ve/bin に配置されています。

- **問題**

  実行形式ファイルがSX-Aurora TSUBASA用かどうか調べる方法はありますか。

- **解決**

  nreadelfコマンドで調べることが可能です。

```
$ /opt/nec/ve/bin/nreadelf -h a.out
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           NEC VE architecture
  Version:                           0x1
  Entry point address:               0x600000004580
  Start of program headers:          64 (bytes into file)
  Start of section headers:          4760248 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         7
  Size of section headers:           64 (bytes)
  Number of section headers:         27
  Section header string table index: 24
```

- **問題**

  VE上で実行中のプロセスの状態を調べる方法はありますか。

- **解決**

  SX-Aurora TSUBASA用の ps コマンドにより、VEで実行中のプロセスの状態を参照することが可能です。

```
$ export -n VE_NODE_NUMBER; /opt/nec/ve/bin/ps -ef
VE Node: 6
UID         PID  PPID  C STIME TTY          TIME CMD
User1     30970     1 75 17:44 ?        00:00:02 ./IMB-MPI1
VE Node: 7
UID         PID  PPID  C STIME TTY          TIME CMD
User1     30977     1 59 17:44 ?        00:00:02 ./IMB-MPI1
VE Node: 5
UID         PID  PPID  C STIME TTY          TIME CMD
User1     30958     1 99 17:44 ?        00:00:02 ./IMB-MPI1
VE Node: 4
UID         PID  PPID  C STIME TTY          TIME CMD
User1     30957     1 99 17:44 ?        00:00:02 ./IMB-MPI1
VE Node: 2
UID         PID  PPID  C STIME TTY          TIME CMD
User1     30919     1  0 17:44 ?        00:00:02 ./IMB-MPI1
VE Node: 3
UID         PID  PPID  C STIME TTY          TIME CMD
User1     30920     1 99 17:44 ?        00:00:02 ./IMB-MPI1
VE Node: 1
UID         PID  PPID  C STIME TTY          TIME CMD
User1     30918     1  0 17:44 ?        00:00:02 ./IMB-MPI1
VE Node: 0
UID         PID  PPID  C STIME TTY          TIME CMD
User1     30917     1  0 17:44 ?        00:00:02 ./IMB-MPI1
```

NQSV 利用の場合は qstat コマンドを使用してください。

```
$/opt/nec/nqsv/bin/qstat
RequestID       ReqName  UserName Queue     Pri STT S   Memory      CPU   Elapse R H M Jobs
--------------- -------- -------- -------- ---- --- - -------- -------- -------- - - - ----
48682.bsv00     run1.sh  user1 batchq          0 RUN -    4.71M     0.00      126 Y Y Y    1
```

- **問題**

  作成したオブジェクトが musl-libc 用か glibc 用か調べる方法はありますか。

- **解決**

  /opt/nec/ve/bin/ve-libc-check スクリプトで判別が可能です。

利用方法は以下の通りです。

```
$ /opt/nec/ve/bin/ve-libc-check ./a.out
  This is compiled with musl-libc: /home/userxxx/a.out
```

上記のようなメッセージが表示される場合、"a.out"はmusl-libcを用いてビルドされています。何も表示されない場合はmusl-libcを用いていません。

注1) musl-libcのサポートは2019年3月末で終了しました。SX-Aurora TSUBASAソフトウェアをご利用中のお客様は、以下の通りglibc環境への移行とプログラムの再コンパイルをお願いします。

- glibcに対応したSX-Aurora TSUBASAソフトウェアにアップデートしてください。
- glibc環境でプログラムを再コンパイルして下さい。

詳細は インストレーションガイドを参照してください。

注2) ".s"ファイルから生成されたオブジェクトファイルを"ve-libc-check"で確認することは出来ません。また、musl-libcでビルドしたライブラリを動的にリンクあるいはロードするVEプログラムが、glibcでコンパイル・リンクされている場合には、"ve-libc-check"で確認することは出来ません。

- **問題**

  どのような環境変数が利用できますか。

- **解決**

  プログラムの実行を制御する環境変数は、たとえば以下があります。

  VE_NODE_NUMBER
  プログラムが実行される VEノード番号を指定します。

  VE_LD_LIBRARY_PATH
  プログラム実行時に検索するライブラリパスを指定します。

  VE_LD_PRELOAD
  動的リンカにプリロードする共有ライブラリのパスを設定します。

- **問題**

  既定のライブラリ検索パスを指定するにはどうしたらよいですか。

- **解決**

  検索パスを追加する場合は、下記のディレクトリに設定ファイル(<任意の文字列>.conf)を作成し、SX-Aurora TSUBASA用の ldconfig を実行します。

  /etc/opt/nec/ve/ld.so.conf.d/

```
(例)
$ cat /etc/opt/nec/ve/ld.so.conf.d/local_lib.conf
/usr/local/local_lib/lib
$ sudo /opt/nec/ve/glibc/sbin/ldconfig
```

- **問題**

  VE プログラムのデバッグに gdb は利用できますか。

- **解決**

  はい。SX-Aurora TSUBASA 用の gdb を提供しています。

付録 A　発行履歴

A.1　発行履歴一覧表

2018 年	5 月	初版
2018 年	12 月	2 版
2019 年	5 月	3 版
2019 年	9 月	4 版
2020 年	1 月	5 版
2020 年	12 月	6 版
2021 年	5 月	7 版

A.2　追加・変更点詳細

4 版

- はしがきを更新
  - VH/VEハイブリッド実行をサポートするNEC MPI バージョンを追加
- 2.2 MPIプログラムのコンパイル方法を更新
  - musl-libc用の説明を削除
  - VH上で動作するMPIプログラムのコンパイル方法を追加
- 3.1.2 MPIプログラムの実行方法を更新
  - VH/VEハイブリッド実行向けの記載を追加・更新
- 3.2.3 MPIプログラムの実行方法を更新
  - VH/VEハイブリッド実行向けの記載を追加・更新
- 5.1 PROGINF機能を更新
  - -proginfオプションの指定を削除
- 5.2 FTRACE機能を更新
  - VE上で動作するMPIプログラムでFTRACE機能が利用可能の記述を追加
- 5.3 プロファイラの使用方法を更新
  - VH上で動作するプログラム用のGMON_OUT_PREFIX環境変数を追加

5 版

- 4.2 高速I/Oを更新

- 高速I/Oを有効にする方法を変更

6 版

- 3.2.3 MPIプログラムの実行方法

- NQSVでの利用例を更新

- 4.1 ScaTeFS ダイレクトI/O

- NQSVでの利用例を更新

- 4.2 高速I/Oを更新

- 利用条件を変更

- 5.1 PROGINF機能

- 出力例を更新

7 版

- 3.3 PBSを利用して、プログラムを実行する場合

- PBSでの利用方法を追加

- 4.1 ScaTeFS ダイレクトI/O

- PBSでの利用例を追加

- 4.2 高速I/Oを更新

- PBSでの利用例を追加

SX-Aurora TSUBASA システムソフトウェア

プログラム実行クイックガイド

2021年 5月 第7版

日本電気株式会社

東京都港区芝五丁目７番１号

TEL(03)3454-1111（大代表）