

TRMM データ利用講習会

第 2 部

TRMM データの読み出し方法 (PR, TMI, VIRS)

Ver 1.01

平成 10 年 12 月 4 日

宇宙開発事業団

地球観測データ解析研究センター

目次

目次	2
1. HDF ライブラリとTSDIS ツールキット	3
1.1 HDF とは?	3
1.2 TSDIS ツールキットとは?	3
2. HDF ライブラリとTSDIS ツールキットのインストール.....	6
2.1 HDF ライブラリのインストール	6
2.1.1 HDF4.1r2(バイナリ)のインストール.....	6
2.1.2 HDF4.0r2(ソース)のインストール.....	7
2.1.3 HDF4 インストール時の注意	9
2.2 TSDIS ツールキットのインストール	9
2.3 環境設定.....	12
3. ツールキットを利用したプログラミング.....	14
3.1 プログラムの流れ.....	14
3.2 C プログラミング	15
3.2.1 プログラム例.....	15
3.2.2 コンパイル.....	17
3.3 F77 プログラミング	18
3.3.1 プログラム例.....	18
3.3.2 コンパイル.....	19
4. TRMM データの可視化ツール.....	21
4.1 HDF 形式に対応するソフトウェア	21
4.2 UNIX 用ソフトウェア.....	21
4.3 パソコン用ソフトウェア	22
4.4 参考ソフトウェア.....	22
付録1. C サンプルプログラム.....	24
2A25 サンプル	24
1B01 サンプル	28
1B11 サンプル	30
3A25 サンプル	32
付録2. F77 サンプルプログラム.....	34
2A25 サンプル	34
1B01 サンプル	38
1B11 サンプル	40
3A25 サンプル	42

1. HDF ライブラリとTSDIS ツールキット

1.1 HDF とは？

TRMM データは HDF (Hierarchical Data Format)フォーマットで作成されています。HDF は、National Center for Supercomputing Applications (NCSA)で開発されたデータ圧縮フォーマットです。HDF フォーマットのデータを C やフォートランのプログラムを使って読むためには、NCSA でフリーで配布している HDF ライブラリを利用する計算機環境にあらかじめインストールしておく必要があります。HDF ライブラリのインストール方法については、第2章で説明します。

表1に HDF の最新バージョン¹ (HDF4.1r2) が対応している機種とコンパイラの情報をまとめます。HDF は UNIX だけでなく、PC や Macintosh についてもサポートされています。また、対応している機種のうち、大部分についてはコンパイル済みのライブラリが配布されています。

TRMM データは HDF4.0r2 のライブラリで作成されていますが、これは現在では古いバージョンになってしまうため、NCSA のサイトから入手できないようです。しかしながら、HDF4.0r2 のソースファイルは NASA のサイトからダウンロードが可能です。EORC では SGI と Linux について HDF4.1r2 をインストールして利用した実績がありますが、特に動作には問題がないようです。

図1に、HDF に関する情報を集めることのできる Web サイトをまとめます。

HDF 一般	http://hdf.ncsa.uiuc.edu/
HDF4.1r2 ダウンロード	http://hdf.ncsa.uiuc.edu/obtain.html
HDF4.0r2 ダウンロード	ftp://daac.gsfc.nasa.gov/software/hdf/HDF_prev_releases
ドキュメント	http://hdf.ncsa.uiuc.edu/doc.html
FAQ	http://hdf.ncsa.uiuc.edu/HDF-FAQ.html

図1 HDF 関連サイト

1.2 TSDIS ツールキットとは？

HDF フォーマットのデータファイルの扱いにはいくつかの方法があります。ファイルに含まれるデータを C やフォートランのプログラムで読み込んで処理をする場合は、HDF ライブラリを使うか、TRMM 用に NASA

¹ 1998/12/1 現在。1998/11/6 に HDF5 Ver.1.0.0 がリリースされたが、TSDIS ツールキットは HDF4 (HDF4.0r2) で構築されているので、ここでは HDF4 についてのみ述べる。

が開発した TSDIS (TRMM Science Data and Information System) ツールキット(内部で HDF ライブラリを呼んでいる)を使うのが便利です。

ツールキットは TRMM および GV (Ground Validation) データを計算機環境の違いに左右されずに、C またはフォートランのプログラムで扱うためのライブラリ群です。特に、データへのアクセスが容易になるように様々なルーチンが用意されています。TRMM の標準プロダクトもそれ自体、ツールキットを利用して作成されています。通常はアルゴリズムの変更などの理由によって、ツールキットのバージョンが上がっていきます。ある TRMM の標準プロダクトがどのバージョンのツールキットを利用して作成されているかという情報は、プロダクトの中のメタデータに含まれています。基本的にはプロダクトを作成したバージョンのツールキットを利用してデータを扱うことが想定されています。しかし、ツールキットのライブラリは上位互換性なので、以前のバージョンのツールキットで作成されているファイルは、それよりも新しいバージョンのツールキットで扱うことが大抵の場合可能です¹。

ツールキットの最新バージョン² (TSDIS Toolkit Release 4.9) がサポートしているプラットフォームは、Sun (SunOS 4.1.4, Solaris 2.5), HP 9000, DEC Alpha, SGI, Linux です。この他にも、FreeBSD にインストール実績があります。ツールキットのインストール方法については、第2章で説明します。ツールキットに関する情報は図2に示した Web サイトで入手可能です。

ツールキット一般	http://www-tsdis.gsfc.nasa.gov/tsdis/tsdistk.html
ダウンロード	http://www-tsdis.gsfc.nasa.gov/cgi-bin/download
Toolkit ユーザーズマニュアル	http://www-tsdis.gsfc.nasa.gov/tsdis/Documents/ICSVol2.pdf
Toolkit クイックリファレンス	http://www-tsdis.gsfc.nasa.gov/tsdis/Documents/ParameterDictionary.pdf

図2 TSDIS ツールキット関連サイト

HDF ライブラリは TRMM データのみならず HDF 全般について利用可能ですが、ツールキットは TRMM の専用になっています。今回はこのツールキットを使ったデータの読み方について、第3章で説明を行います。HDF ライブラリのみを利用したデータの処理については、HDF の Web サイトにあるドキュメントやサンプルプログラムを参考にして下さい。なお、HDF フォーマットのデータを可視化するためのソフトウェアについては、第4章で簡単に説明します。

なお、ツールキットのマニュアルおよびクイックリファレンスの記述内容は、必ずしも最新版のアルゴリズムに対応していない場合もあります。その場合は、第1部で紹介した NASA/TSDIS 配布のフォーマット説明書の最新版を参考にするようにして下さい。

¹ ワーニングが出たり (無視できる)、読み込みに失敗する (無視できない) こともある。プロダクト ID の異なる TRMM データを扱う場合は、古いバージョンのツールキットも残しておくことをお勧めする。

² 1998/12/1 現在。

³ この他に、ツールキットの中に含まれている include ファイル (IO_PR.h, IO_TMI.h など) 中の記述も参考になる。

表 1 HDF4.1r2 でサポートされるプラットフォーム

Platform(OS)	C-Compiler	Fortran-Compiler
Sun4(SunOS 4.1.4)	GCC 2.6.3	f77 SC1.0
Sun4(Solaris 2.5)	CC SC4.0	f77 SC4.0
SGI-Indy(IRIX 5.3)	CC 3.19	f77
SGI-Indy(IRIX v6.2)	CC	f77 6.2
SGI-Origin(IRIX64 v6.4-n32)	CC 7.20	f77 7.20
SGI-Origin(IRIX64 v6.4-64)	CC 7.20	f77 7.20
HP9000/735(HP-UX 9.03)	CC A.09.84	f77 09.16
HP9000/755(HP-UX B.10.20)	CC A.10.32.03	f77
Exemplar(HP-UX A.09.03)	CC 6.5	fc 9.5
Cray T90		
CFP (UNICOS 10.0.0bt u10.13)	CC 5.0.5.0	f90 3.0.2.0
IEEE (UNICOS 10.0.0bt d10.25)	CC 5.0.5.0	f90 3.0.2.0
Cray C90 (UNICOS 803.2)	<i>Not Tested</i>	<i>Not Tested</i>
IBM SP2 (single node, v4.2.1)	XLC 3.1.4.0	f77 4.1.0.6
DEC Alpha/Digital Unix v4.0	CC	f77
DEC Alpha/OpenVMS AXP v6.2	DEC C 5.0	Digital Fortran 77 V7.1
DEC Alpha/OpenVMS AXP v7.1	DEC C 5.5	DEC Fortran v6.3
VAX OpenVMS v6.2	DEC C 5.2	DEC Fortran v6.3
IBM PC - Intel Pentium		
Solarisx86 (2.5.1)	GCC 2.7.2.1	<i>Not Tested</i>
Linux (elf) (2.0.30)	GCC 2.7.2.1	fort77 (f2c) 2.7.2.1
FreeBSD (2.2.1)	GCC 2.7.2	f77 2.7.2.1
PowerPC(Mac-OS-7.6)	MetroWerks	<i>Not Tested</i>
	Codewarrior (CW3)	
68k-far/4i/8d(Mac-OS-7.6)	<i>Not Tested</i>	<i>Not Tested</i>
Windows NT/95	MSVC++ 5.0	Visual Fortran 5.0
DEC Alpha NT	MSVC++ 5.0	Visual Fortran 5.0
T3E (unicosmk 2.0.2.16)	CC 6.0.2.0.5	f90 3.0.2.0

注) 表中でプラットフォームとコンパイラが記述されているものについては、NCSAによってHDFがテスト済みであり、コンパイル済みのバイナリデータが提供されています。表にあるプラットフォームのうち、Cとフォートラン・コンパイラの列に"Not Tested"と記述されているものは、プラットフォームはサポートしているものの、まだテストがされていないものです。

2. HDF ライブラリと TSDIS ツールキットのインストール

2.1 HDF ライブラリのインストール

HDF のインストール方法としてお薦めな方法は、HDF4.1r2 のコンパイル済みのバイナリ・データをインストールする方法と、HDF4.0r2 のソースコードをコンパイルする方法の2通りのやり方があります。ツールキットはHDF4.0r2をベースに作成されていますが、HDF4.1r2ではバグが解消されており、さらに、主な計算機環境およびOSについてNCSAからコンパイル済みのファイルが提供されています。しかし、計算機環境によってはコンパイル済みのバイナリでは不都合もありますので、その場合にはHDF4.0r2をソースからコンパイルしてインストールするのが適当でしょう。以下では、HDF4.1r2のバイナリのインストールと、HDF4.0r2のソースからのコンパイル方法について説明します。Linux または SGI へインストールする場合は、2.1.3 節の注意を参照して下さい。

2.1.1 HDF4.1r2(バイナリ)のインストール

NCSA のサイト(もしくはミラーサイト)から、自分の計算機環境および OS のバージョンに対応したHDF4.1r2のコンパイル済みファイル(pre-compiled binary)をダウンロードします。HDF は個人で使う場合は必ずしもスーパーユーザーの権限は必要ありません。自分のディレクトリ下にインストールして、読み込みプログラムのコンパイル時にそこにリンクを張れば使うことができます。他の人と共有したい場合は、ファイルを展開後、展開したディレクトリ構造ごと、マシンの管理者に root の下のディレクトリにコピーしてもらうとよいでしょう。

ダウンロードは、Web ツール(Netscape や Internet Explorer)を使う場合は、URL として

`ftp://ftp.ncsa.uiuc.edu/HDF/HDF_Current/bin/`

を入力します。そこからさらに自分の計算機環境および OS のバージョンに対応したディレクトリに移動し、自分の計算機のインストールしたいディレクトリに目的のファイルをダウンロードします。

一方、anonymous ftp を使う場合は、以下のように NCSA のアドレスを入力します。

`% ftp ftp.ncsa.uiuc.edu`

繋がらない場合は、IP アドレスを直接入力する。

`% ftp 141.142.3.77`

NCSA に繋がった後は、通常の anonymous ftp と同様にして、アカウントとして anonymous、パスワードとして自分のメールアドレスを入力します。その後、該当のディレクトリに移動します。例えば [Solaris マシン](#) にインストールをしたい場合には、以下のようにして移動し、目的のファイルを自分のマシンにダウンロードします。

```
ftp> cd /HDF/HDF_Current/bin/solaris
ftp> bin
ftp> get 4.1r2\_solaris.tar.gz
ftp> bye
```

ftp を終了した後に、自分のマシンのインストールしたいディレクトリの下で、ダウンロードしたファイルの解凍展開をして下さい。

```
% gzip -cd 4.1r2\_solaris.tar.gz | tar xvf -
```

すると、現在いるディレクトリの下に [HDF4.1r2_solaris](#) というディレクトリができます。この時のディレクトリ構造は

HDF4.1r2_solaris /COPYING	: Copyright
/README	: 簡単な使い方
/bin/	: HDF のユーティリティ (ツール) のディレクトリ
/include/	: インクルードファイルのディレクトリ
/lib/	: ライブラリのディレクトリ
/man/	: ツールのマニュアルのディレクトリ
/release_note/	: HDF ライブラリの説明のディレクトリ

のようになっています。これでインストールは完了です。他の場所にインストールする場合は、[HDF4.1r2_solaris](#) のディレクトリごと、目的先のディレクトリにコピーして下さい。

```
% cp -r HDF4.1r2\_solaris <インストール先>
```

2.1.2 HDF4.0r2(ソース)のインストール

コンパイル済みの HDF4.1r2 が利用できないなどの場合には、HDF4.0r2 のソースをコンパイルして利用するのがよいでしょう。HDF4.1r2 のソースをコンパイルしても構わないのですが、HDF4.0r2 に比べるとコンパイルがかなり複雑になっています。

ライブラリのコンパイルには、ANSI C コンパイラが必要です。ANSI C が使えないプラットフォームでは、フリーの GNU ANSI コンパイラ gcc を使っています。HDF のインストールがうまくいかない場合、C コンパイラを gcc にするとうまくいくことがあります。

HDF4.0r2 は図 1 に示したサイトからダウンロードが可能です。HDF4.1r2 と同様にして、Web 経由でも anonymous ftp 経由でも持ってくるすることができます。

インストール先のディレクトリに HDF4.0r2.tar.Z をダウンロードした後、ファイルを解凍展開します。

```
% zcat HDF4.0r2.tar.Z | tar xvf -
```

すると、現在のディレクトリの下に HDF4.0r2 というディレクトリができます。この下のディレクトリ構造は以下ようになります。

HDF4.0r2/COPYING	
/INSTALL	:インストール手順の説明(必見)
/Makefile.in	
/README	:ディレクトリの説明など(必見)
/config.guess	
/config.sub	
/config/	:マシンごとの Makefile のディレクトリ
/configure	:マシンごとの Makefile をつくる configure(必須)
/configure.in	
/hdf/	:HDF のソースコードのディレクトリ
/install-sh	
/man/	:HDF のマニュアルのディレクトリ
/mfhdf/	:netCDF のディレクトリ
/mkinstalldirs	
/move-if-change	
/release_note/	:HDF ライブラリの説明のディレクトリ
/win32mak.zip.uu	

コンパイルの前に、configure を使って、インストールしたい計算機に対応した Makefile を作成します。CC や CFLAG などのデフォルトの値を変えたい場合は、HDF4.0r2/config/mh-[<OS>](#) (例えば OS が Solaris2.5 ならば mh-solaris) の中の設定を書き換えます。

configure を実行する際、デフォルトでは最終的なライブラリのインストール先は /usr/local です。その下に、ライブラリ、ユーティリティ、マニュアル、インクルードファイルなどが、/usr/local/lib、/usr/local/bin、/usr/local/man、/usr/local/include の下に書き込まれます。しかし、以前からあるライブラリなど(例えば libjpeg.a はすでに /usr/local/lib にインストールされている可能性がある)を上書きするのを避けるために、別にディレクトリを作っておいてインストールするのがよいでしょう。ここでは、自分のホーム下の HDF4.0r2 というディレクトリにインストールする例を示します。

```
% ./configure -v --prefix=/home/trmm11/work/HDF4.0r2
```

インストール先のディレクトリは prefix の後で指定しています。これを実行すると、それぞれの計算機環境にあった Makefile が自動的に作成されます。ライブラリのコンパイルには

```
% make
```

として下さい。コンパイルには少し時間がかかります。ライブラリがうまくコンパイルされたかどうかの確認は、コンパイル終了後、

```
% make test
```

とすると、一通りのテストが行えます。結果が標準出力に出てきますので、

```
% make test >& make.test.out
```

のように、アウトプットをファイルにセーブしておいて確認すると便利です。上のアウトプットの中身を確認して問題がないようでしたら

```
% make install
```

として、さきほど設定したディレクトリにライブラリをインストールして下さい。

2.1.3 HDF4 インストール時の注意

- Linuxへのインストール上の注意

HDF4 の Makefile が高機能であるため、CPU が違うとインストールできないケースがでてきます。

- (1) HDF4.0r2 は、Pentium2 など CPU コード「i686」のマシンへは Makefile を修正しないとインストールできません。MMX Pentium など CPU コード「i586」や「i486」のマシンには Makefile の修正なしにインストール可能です。
- (2) HDF4.1r2 は、CPU コード「i486」～「i686」では、Makefile の修正なしにインストール可能です。
- (3) RedHat のLinuxにインストールする際には、HDF のホームページ上に掲載されている「インストール上の注意」を参照してください。Slacware および S.u.S.E.のLinuxでは問題ありません。

- SGI (IRIX) へのインストール上の注意

HDF4 の Makefile が高機能であるため、OSのバージョンとライブラリのバージョンに食い違いがあるとインストールできないケースがでてきます。

- (1) HDF4.0r2 は問題なくインストールすることができます。
- (2) HDF4.1r2 は、OS のバージョンとライブラリのバージョンに食い違いがある場合は、Makefile を修正しないと正しいロードモジュールを作成することができません。

2.2 TSDIS ツールキットのインストール

TSDIS ツールキットは、図2に示したダウンロードサイトから、Web ツールを使って取ってくることができます。その際に、名前とEメール・アドレスの入力を求められるので、入力してから、Download Files のボタンをクリックして下さい。

図3の通り、TSDIS ツールキットは現在、Release 4.7¹と Release 4.9²が入手可能です。上位互換ですので、できるだけ新しいリリースのものをインストールするようにしましょう。Toolkit Installation Guide は、Toolkit の Distribution の中に、例えば、リリース 4.9 の場合は、INSTALL.R49 の名前で含まれています。Topographic and Land/Sea Data (標高、海陸データ) は別にダウンロードする必要があります。このデータはツールキットのバージョンアップとともに更新されていますので、新しいリリースのツールキットをダウンロードする際には必ず一緒にダウンロードするようにして下さい。

¹ 1998/5/6 リリース。TRMM 標準プロダクトに対応(1998/6/1～1998/8/31 リリースのレベル1プロダクト)。

² 1998/10/8 リリース。TRMM 標準プロダクトおよび TRMM リアルタイムプロダクトに対応(1998/9/1～現在リリースの全プロダクト)。なお、リリース 4.8 は欠番である。

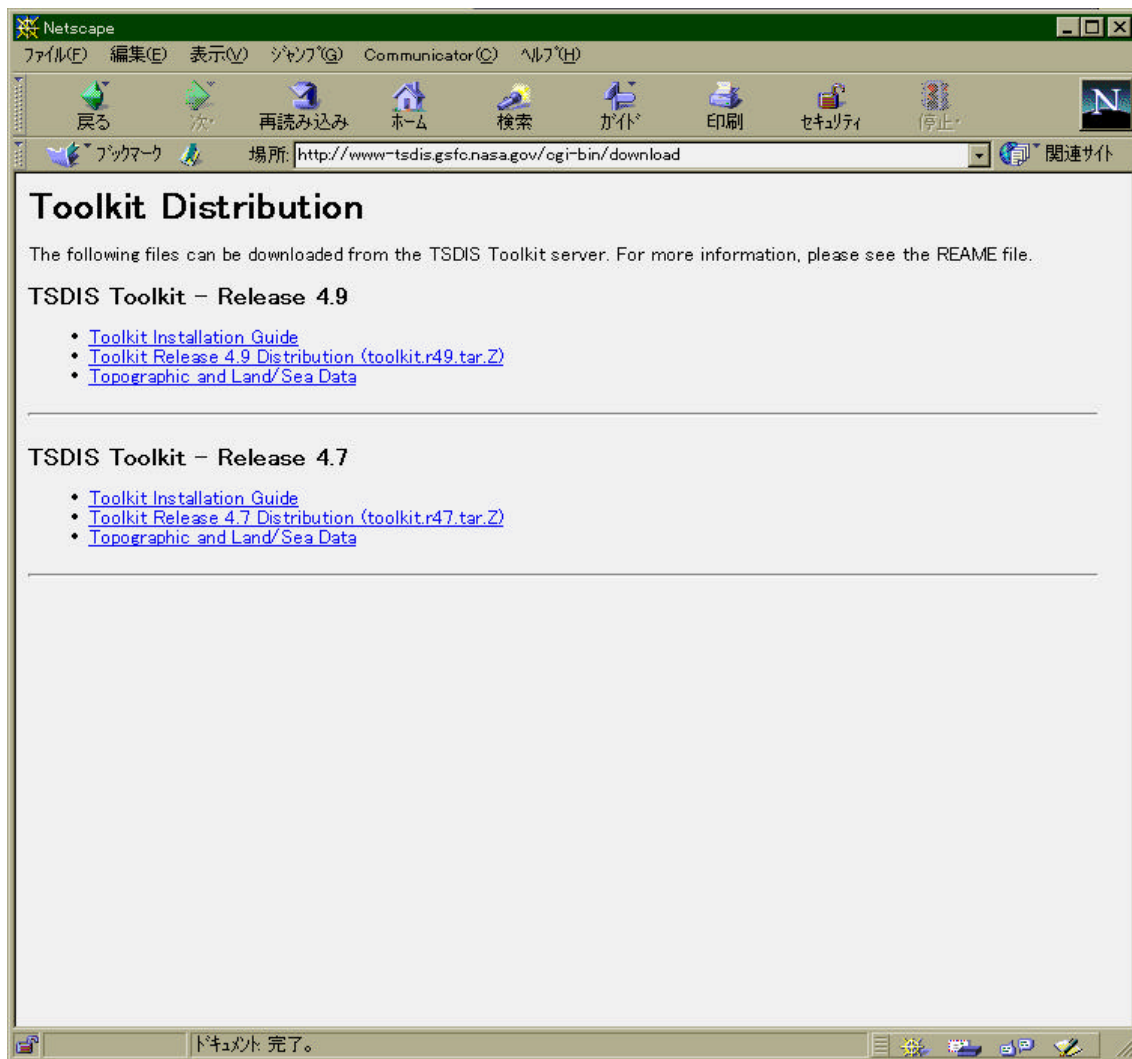


図3 TSDIS ツールキット・ダウンロード画面

このようにダウンロードしたファイルを解凍展開します。

```
% zcat toolkit_r49.tar.Z | tar xvf -
% zcat tsdistk_data.tar.Z | tar xvf -
```

現在のディレクトリの下に `toolkit_4.9` というディレクトリと `data` というディレクトリができますので、`data` 以下の標高・海陸データを `toolkit_4.9` のディレクトリの下に移動します。

```
% mv data toolkit_4.9/
```

この時 [toolkit_4.9](#) 以下のディレクトリ構造は以下のようになります。

toolkit_4.9 /CHANGELOG	
/COMPILING	: ツールキット・ライブラリの使い方の説明
/INSTALL.R49	: インストール方法の説明 (必見)
/MAKE.ALPHA	: DECAalpha 用の Makefile
/MAKE.HP9	: HP 用の Makefile
/MAKE.LINUX	: Linux 用の Makefile
/MAKE.SGI	: SGI 用の Makefile
/MAKE.SOLARIS25	: Solaris2.5 用の Makefile
/MAKE.SUNOS414	: SUNOS4.1.4 用の Makefile
/NASDA_INSTALL	: TSDIS 以外の環境でのインストール
/README	: ディレクトリの説明 (必見)
/RELEASE.NOTES	: ヒストリー
/TESTING.MATRIX	
/config	: configuration ファイルのディレクトリ
/data	: 標高 (etop05.dat) 海陸情報 (dbglobe93.grd) データのディレクトリ
/db	: TSDIS ASCII データベースのディレクトリ
/include	: include ファイルのディレクトリ
/lib	: ライブラリのインストール先ディレクトリ
/src	: ツールキットのソースコードのディレクトリ

インストール作業を始める前に、自分の計算機環境に対応した MAKE.<OS> (例えば Solaris マシンの場合は MAKE.SOLARIS25 になる) を Makefile という名前にコピーします。ここでは Solaris の例を示します。

```
% cp MAKE.SOLARIS25 Makefile
```

コピーしてできた Makefile の中の HDFINC のパス (HDF ライブラリの中の include ファイルのパス) の設定を修正します。例えば、HDF ライブラリが [/home/trmm11/work/HDF4.1r2_solaris/](#) にインストールされている場合は

```
#####  
#      Please modify the following path as described above.  
#####  
  
HDFINC = /data/HDF/HDF4.0r2/hdf/include
```

上に記述されているパスを

```
HDFINC = /home/trmm11/work/HDF4.1r2_solaris/include
```

のように書き直します。

また、NASA/TSDIS で使われているデータベース(ツールキットに ASCII のデータベースとして含まれている)を自分の計算機環境で使えるように、コンパイル・オプションの設定(CFLAGS, FFLAGS)を修正します。

```
#Define the C compiler, flags and compiling options:
#   -DPSIZE_64 for 64-bit machine environment
#   -DTSDIS_TK_ENV for TSDIS environment
#   -DOPERATION_ENV for operation environment
#   -DTSU_ENV for TSU environment
#   -DNASDA_ENV for using the ASCII database
CC = cc
CFLAGS= -w -Xa $(DEBUG_FLAG) -D$(MACHINE) -DX_WCHAR -DTSU_ENV

#Define the F77 compiler and flags
FC = f77
FFLAGS = $(DEBUG_FLAG) -D$(MACHINE) -DLANGUAGE_FORTRAN -DTSU_ENV
```

CFLAGS と FFLAGS について、オプション、**-DNASDA_ENV** を書き加えます。

```
CFLAGS= -w -Xa $(DEBUG_FLAG) -D$(MACHINE) -DX_WCHAR -DTSU_ENV -DNASDA_ENV

FFLAGS = $(DEBUG_FLAG) -D$(MACHINE) -DLANGUAGE_FORTRAN -DTSU_ENV -DNASDA_ENV
```

これでコンパイル準備完了です。

```
% make
```

として、コンパイルをします。成功すると、`/home/trmm11/work/toolkit_4.9/lib/`の下に、`libtsdistk.a` というライブラリができあがります。C やフォートランのプログラム中で TSDIS ツールキットを使う場合には、このライブラリの他に、include ファイル(`toolkit_4.9/include/`)やデータベース(`toolkit_4.9/data/`, `toolkit_4.9/db/`)などを使いますので、共有のスペースに置く場合は `toolkit_4.9` のディレクトリごとコピーするほうがよいでしょう

```
% cp -r toolkit_4.9 <インストール先>
```

2.3 環境設定

HDF ライブラリを利用する場合には、それぞれのユーザは自分の.cshrc ファイルに、使用する HDF ライブラリのバージョンの環境変数を設定する必要があります。例えば、`/home/trmm11/work/HDF4.0r2` を利用する場合は以下ようになります。

```
set path=(/home/trmm11/work/HDF4.0r2/bin/ $path)
setenv LD_LIBRARY_PATH /home/trmm11/work/HDF4.0r2/lib:$LD_LIBRARY_PATH
setenv HDFINC /home/trmm11/work/HDF4.0r2/include
```

また、TSDIS ツールキットを利用する場合にも、やはりそれぞれのユーザの.cshrc ファイルに環境変数を設定する必要があります。例えば [/home/trmm11/work/toolkit_4.9](#) を利用する場合は以下のようになります。

```
setenv LD_LIBRARY_PATH /home/trmm11/work/toolkit_4.9/lib:$LD_LIBRARY_PATH
setenv TSDISTK /home/trmm11/work/toolkit_4.9
```

以上で、HDF ライブラリと TSDIS ツールキットの利用準備が完了です。

3. ツールキットを利用したプログラミング

3.1 プログラムの流れ

TRMM データを TSDIS ツールキットを使って C やフォートラン(F77) のプログラムで読み込むには、図4のような流れでプログラムを作成します。

ヘッダーファイルの記述

プログラムで利用するツールキット用のヘッダー情報ファイル(include ファイル)を記述します。全プロダクト共通の入出力の include ファイルやセンサに固有のものなどがあります。

入出力構造体の宣言

上のヘッダーファイル(include ファイル)に記述されているデータの**入出力構造体**¹を任意の名前に宣言します。以降、プログラム中では宣言された名前で参照されます。

HDF ファイルのオープン

読み込みたい HDF ファイルをオープンします。

メタデータの読み込み

HDF データのうち、**コアメタデータ**、**PS メタデータ**²の各要素を変数に読み込みます。

スキャン毎のデータの読み込み

TRMM のレベル 1 および 2 プロダクトはスキャン毎に観測データが入っています。これらのサイエンス・データ(**SDS**³)は各スキャン毎に読み込みます。各センサ、プロダクトに含まれるサイエンス・データの詳しい構成については、本講習会テキスト Appendix または、NASA/TSDIS 配布のフォーマット説明書、ツールキット・マニュアルなどを参照してください。

(注意) 一部の SDS データについて Appendix 中で、実数のデータを整数値に直して格納するために定数を引いたり掛けたりしていることが記述されています。しかし、ツールキットではたいていの場合はこの操作を読み込みの際に自動的に行っていますので(していない場合もあります)注意して扱って下さい。Parameter Dictionary におけるフォーマットの記述も参考になるでしょう。

格子データの読み込み

TRMM のレベル 3 プロダクトは格子データの形になっていて、SDS は一度に全部読み込みます。

HDF ファイルのクローズ

HDF ファイルをクローズし、データの操作を終了します。

¹ 別冊の Appendix 参照

² 別冊の Appendix 参照

³ 別冊の Appendix 参照

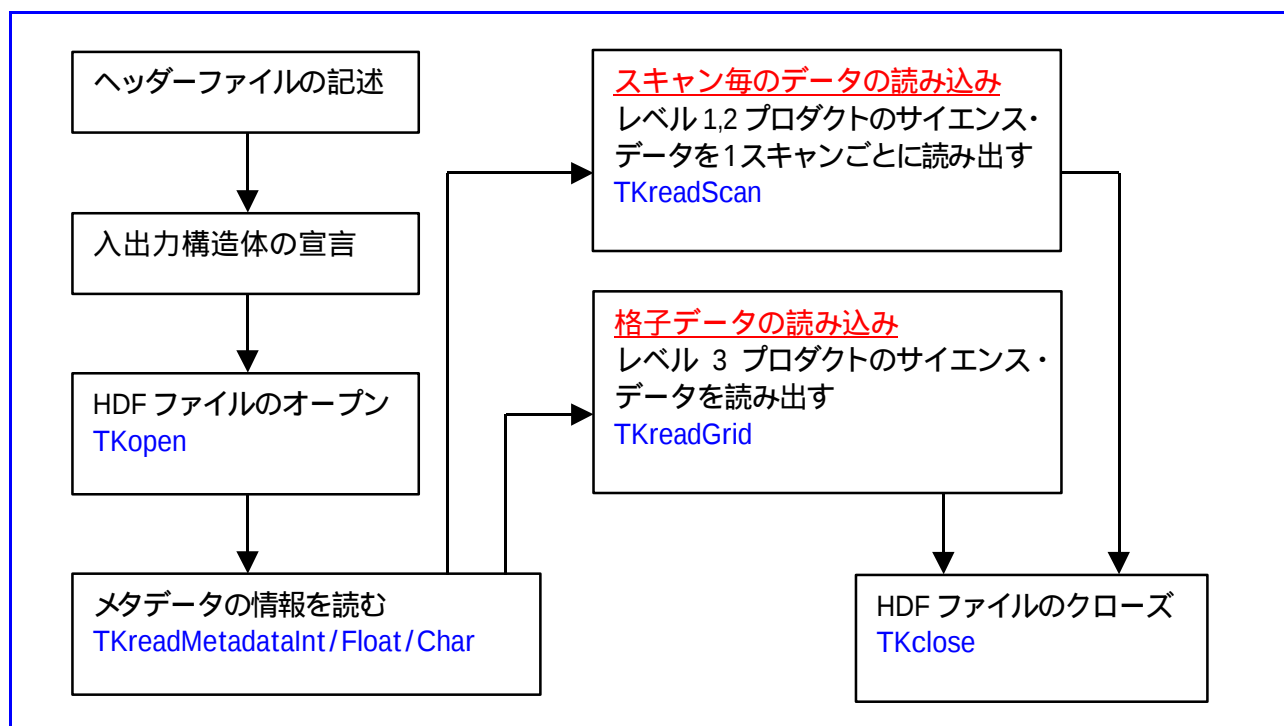


図4 ツールキットを利用したプログラムの流れの概念図

3.2 C プログラミング

C 言語のサンプルプログラムは、付録 1 に添付してあります。ここでは、TRMM の標準プロダクトのうち 2A25 (PR)、3A25(PR)、1B01(VIRS)、1B11(TMI)についての簡単な読み込みプログラムをまとめました。これらのプログラムは、EORC の TRMM の Web サイト(<http://www.eorc.nasda.go.jp/TRMM/>)の中での FAQ のページからもダウンロードすることが可能です。

3.2.1 プログラム例

図 4 のツールキットによるプログラムの流れに従って、C 言語によるプログラムの仕方を PR の 2A25 プロダクトの読み込み(格子データの読み込みについては、3A25 プロダクト)を例に取って、簡単に説明します。

ヘッダーファイルの記述

```
#include <IO.h>
#include <IO_PR.h>
#include <IO_INTR_PR.h>
```

必須

例えば TMI の場合は <IO_TMI.h>

例えば TMI の場合は <IO_INTR_TMI.h>

入出力構造体データの宣言

```
IO_HANDLE granuleHandle2A25;
L2A_25_SWATHDATA L2A25_data;
```

入出力構造体の宣言

スキャン毎のデータの構造体の宣言

左側はヘッダーファイルに定義されている名称

右側はユーザが定義する任意の名称

¹ 標準プロダクト名については、第1部表 2-2 を参照。

HDF ファイルのオープン

```
status = TKopen (HDF ファイル名, TK_L2A_25, TK_READ_ONLY, &granuleHandle2A25);
```

一番目の引数は HDF ファイル名
二番目の引数はデータタイプの指定
三番目の引数はオープン条件の指定
四番目の引数は IO_HANDLE でユーザが定義した入出力構造体名

メタデータの読み込み

```
status = TKreadMetadataInt ( &granuleHandle2A25, TK_ORBIT_SIZE, &numberOfScan );
```

整数

```
status = TKreadMetadataFloat ( &granuleHandle2A25, TK_FILE_SIZE, &fileSize );
```

実数

```
status = TKreadMetadataChar ( &granuleHandle2A25, TK_GRANULE_ID, &granuleID );
```

文字

要素によって、Int, Float, Char のどれを使うかを区別

一番目の引数はユーザ定義の入出力構造体宣言名

二番目の引数はメタデータの要素指定 (第1部 Appendix 参照)

三番目の引数はユーザ定義の変数名

スキャン毎のデータの読み込み (レベル1, 2 データ)

```
for ( iScan=1; iScan <= numberOfScan; iScan++ ) {
```

```
    status = TKreadScan ( &granuleHandle2A25, &L2A25_data );
```

一番目の引数はユーザ定義の入出力構造体宣言名

二番目の引数はスキャン毎のデータの構造体宣言名

```
    printf( "Lat, Lon : %f,%f ¥n", L2A25_data.geolocation[24][0], L2A25_data.geolocation[24][1] );
```

```
}
```

スキャン毎のデータの構造体名の引用はメンバ名を

ドットで区切って指定する

上の例では、クロストラック方向の 25 番目のピンの緯度および経度情報を書き出している

格子データの読み込み (レベル3 データ)

```
status = TKreadGrid ( &granuleHandle3A25, &L3A25_data );
```

一番目の引数はユーザ定義の入出力構造体宣言名

二番目の引数は格子データの構造体宣言名

```
printf( "RainMean : %f¥n", L3A25_data.grid1.rainMean[0][71][15]);
```

格子データの構造体名の引用はメンバ名をドットで区切って指定する

上の例では、Grid1(5 度×5 度格子)の格子データについて、高度方向 1 番目、経度方向 72 番目、緯度方向 16 番目の平均降雨強度情報を書き出している

ファイルのクローズ

```
status = TKclose( &granuleHandle2A25);
```

引数はオープンした際の入出力構造体宣言名

3.2.2 コンパイル

ここでは NASDA/EORC の計算機環境でテストを行った SUN、SGI、DEC、HP、Linux における C プログラムのコンパイルの例を示します。以下の例ではすべて、付録 1 に掲載されているサンプルプログラム `c_2a25rd.c` (実行形式は `c_2a25rd`) のコンパイルを想定しています。

- SUN (Solaris2.5)

コンパイル・オプションとして、`-DSUN -Xc -lnsl` をつける。

```
cc -DSUN -Xc -o c_2a25rd c_2a25rd.c ¥  
    -I/home/trmm11/work/HDF4.0r2/include -I/home/trmm11/work/toolkit_4.9/include ¥  
    -L/home/trmm11/work/HDF4.0r2/lib -L/home/trmm11/work/toolkit_4.9/lib ¥  
    -ltsdstk -lmfhdf -ldf -ljpeg -lz -lm -lnsl
```

- SGI (IRIX6.4)

1) コンパイル・オプションとして、`-DSGI -xansi -fullwarn` をつける

```
cc -DSGI -xansi -fullwarn -o c_2a25rd c_2a25rd.c ¥  
    -I/home/trmm11/work/HDF4.0r2/include -I/home/trmm11/work/toolkit_4.9/include ¥  
    -L/home/trmm11/work/HDF4.0r2/lib -L/home/trmm11/work/toolkit_4.9/lib ¥  
    -ltsdstk -lmfhdf -ldf -ljpeg -lz -lm
```

- DEC Alpha

1) コンパイル・オプションとして、`-DDEC_ALPHA -ieee_with_no_inexact -std1` をつける。

```
cc -DDEC_ALPHA -ieee_with_no_inexact -std1 -o c_2a25rd c_2a25rd.c ¥  
    -I/home/trmm11/work/HDF4.0r2/include -I/home/trmm11/work/toolkit_4.9/include ¥  
    -L/home/trmm11/work/HDF4.0r2/lib -L/home/trmm11/work/toolkit_4.9/lib ¥  
    -ltsdstk -lmfhdf -ldf -ljpeg -lz -lm
```

- HP (HP-UX10.20)

1) コンパイル・オプションとして、`-DHP9000 -Aa` をつける。

```
cc -DHP9000 -Aa -o c_2a25rd c_2a25rd.c ¥  
    -I/home/trmm11/work/HDF4.0r2/include -I/home/trmm11/work/toolkit_4.9/include ¥  
    -L/home/trmm11/work/HDF4.0r2/lib -L/home/trmm11/work/toolkit_4.9/lib ¥  
    -ltsdstk -lmfhdf -ldf -ljpeg -lz -lm
```

- Linux

1) コンパイラとして gcc を用いる¹。

2) コンパイル・オプションとして、`-fPIC -DLinux` をつける。

```
gcc -fPIC -DLinux -o c_2a25rd c_2a25rd.c ¥  
    -I/home/trmm11/work/HDF4.0r2/include -I/home/trmm11/work/toolkit_4.9/include ¥  
    -L/home/trmm11/work/HDF4.0r2/lib -L/home/trmm11/work/toolkit_4.9/lib ¥  
    -ltsdstk -lmfhdf -ldf -ljpeg -lz -lm
```

¹ ANSI C が使えない機種では、フリーの GNU ANSI コンパイラ gcc を使っている。

3.3 F77 プログラミング

F77 のサンプルプログラムは、付録 2 に添付してあります。ここでは TRMM の標準プロダクトのうち、2A25(PR)、3A25(PR)、1B01(VIRS)、1B11(TMI)についての簡単な読み込みプログラムをまとめました。これらのプログラムは EORC の TRMM の Web サイト(URL は <http://www.eorc.nasda.go.jp/TRMM/>) 中の FAQ のページからダウンロードすることが可能です。なお、フォートランによるプログラムのソースコードの拡張子は *.f ではなく、必ず *.F とします。これは、プリプロセッサを通すためです。

3.3.1 プログラム例

ヘッダーファイルの記述

#include "TKfortranDeclare.h"	必須
#include "IO.h"	必須
#include "IO_PR.h"	例えば TMI の場合は <IO_TMI.h>

入出力構造体データの宣言

record /WRAPPER_HANDLE/ granuleHandle2A25	入出力構造体の宣言
	F77 では IO_HANDLE ではなく WRAPPER_HANDLE
record /L2A_25_SWATHDATA/ L2A25_data	スキャン毎のデータの構造体宣言
	/WRAPPER_HANDLE/などは、ヘッダーファイルに定義されている名称
	右側はユーザが定義する任意の名称

ファイルのオープン

status = TKopen (HDF ファイル名, TK_L2A_25, TK_READ_ONLY, granuleHandle2A25)	
	一番目の引数は HDF ファイル名
	二番目の引数はデータタイプの指定
	三番目の引数はオープン条件の指定
	四番目の引数はユーザ定義の入出力構造体宣言名

メタデータの読み込み

status = TKreadMetadataInt (granuleHandle2A25, TK_ORBIT_SIZE, numberOfScan)	
status = TKreadMetadataFloat (granuleHandle2A25, TK_FILE_SIZE, fileSize)	
status = TKreadMetadataChar (granuleHandle2A25, TK_GRANULE_ID, granuleID)	
	要素によって、Int, Float, Char のどれを使うかを区別
	一番目の引数はユーザ定義の入出力構造体宣言名
	二番目の引数はメタデータの要素指定
	三番目の引数はユーザ定義の変数名

スキャン毎のデータの読み込み (レベル 1, 2 データのみ)

```
do 100 iScan = 1, numberOfScan
  status = TKreadScan( granuleHandle2A25, L2A25_data )
  write (6,*) 'Lat, Lon : ', L2A25_data.geolocation(1,25), L2A25_data.geolocation(2,25)
100 continue
```

一番目の引数はユーザ定義の入出力構造体宣言名
二番目の引数はスキャン毎のデータの構造体宣言名

スキャン毎のデータの構造体名の引用はメンバ名を
ドットで区切って指定する
上の例では、クロストラック方向の 25 番目のピンの緯
度および経度情報を書き出している

格子データの読み込み (レベル 3 データのみ)

```
status = TKreadGrid ( granuleHandle3A25, L3A25_data )
write (6,*) 'RainMean : ', L3A25_data.grid1.rainMean(16,72,1)
```

一番目の引数はユーザ定義の入出力構造体宣言名
二番目の引数は格子データの構造体宣言名

格子データの構造体名の引用はメンバ名をドットで
区切って指定する
上の例では、Grid1(5 度×5 度格子)の格子データに
ついて、高度方向 1 番目、経度方向 72 番目、緯度方
向 16 番目の平均降雨強度情報を書き出している

ファイルのクローズ

```
status = TKclose (granuleHandle2A25)
```

引数はオープンした入出力構造体の宣言名

3.3.2 コンパイル

ここでは NASDA/EORC の計算機環境でテストを行った、SUN、SGI、DEC、HP におけるフォートラン・プログラムのコンパイルの例を示します。以下の例ではすべて、付録 2 に掲載されているサンプルプログラム f_2a25rd.F(実行形式は f_2a25rd) のコンパイルを想定しています。

- SUN(Solaris2.5)

- 1) コンパイル・オプションとして、**-DLANGUAGE_FORTRAN -lnsl** をつける。

```
f77 -DLANGUAGE_FORTRAN -o f_2a25rd f_2a25rd.F ¥
    -I/home/trmm11/work/HDF4.0r2/include -I/home/trmm11/work/toolkit_4.9/include ¥
    -L/home/trmm11/work/HDF4.0r2/lib -L/home/trmm11/work/toolkit_4.9/lib ¥
    -ltsdtk -lmfhdf -ldf -ljpeg -lz -lnsl
```

- SGI (IRIX 6.4)

- 1) コンパイル・オプションとして、**-DLANGUAGE_FORTRAN** をつける。

```
f77 -DLANGUAGE_FORTRAN -o f_2a25rd f_2a25rd.F ¥
    -I/home/trmm11/work/HDF4.0r2/include -I/home/trmm11/work/toolkit_4.9/include ¥
    -L/home/trmm11/work/HDF4.0r2/lib -L/home/trmm11/work/toolkit_4.9/lib ¥
    -ltsdtk -lmfhdf -ldf -ljpeg -lz
```

- DEC Alpha

- 1) コンパイル・オプションとして、`-DLANGUAGE_FORTRAN -ieee_with_no_inexact -std1 -DPSIZE_64`をつける。

```
f77 -DLANGUAGE_FORTRAN -DDEC_ALPHA -ieee_with_no_inexact -std1 -DPSIZE_64 ¥
-o f_2a25rd f_2a25rd.F ¥
-I/home/trmm11/work/HDF4.0r2/include -I/home/trmm11/work/toolkit_4.9/include ¥
-L/home/trmm11/work/HDF4.0r2/lib -L/home/trmm11/work/toolkit_4.7/lib ¥
-ltsdstk -lmfhdf -ldf -ljpeg -lz
```

- HP (HP-UX 10.20)

- 1) フォートラン・コンパイラは `fort77` を用いる。
- 2) コンパイル・オプションとして、`-DLANGUAGE_FORTRAN -lm` をつける。
- 3) ソースコードの先頭に `program main` を付加する。
- 4) 付録 2 のサンプルプログラムの中の コマンドラインの引数を読み込むルーチン `getarg` を `igetarg` に変更し、3 番目の引数として文字列の長さを指定する。

```
fort77 -DLANGUAGE_FORTRAN -o f_2a25rd f_2a25rd.F ¥
-I/home/trmm11/work/HDF4.0r2/include -I/home/trmm11/work/toolkit_4.9/include ¥
-L/home/trmm11/work/HDF4.0r2/lib -L/home/trmm11/work/toolkit_4.9/lib ¥
-ltsdstk -lmfhdf -ldf -ljpeg -lz -lm
```

4. TRMM データの可視化ツール

4.1 HDF 形式に対応するソフトウェア

TRMM のプロダクトを、第 3 章で説明したようにツールキットを使ってプログラムを通して読み込むのではなく、直接コンピュータ上で画像化するには、HDF 形式に対応している可視化ツールを利用する必要があります。HDF 形式はかなり一般的になってきていますし、特に衛星データには HDF 形式で書かれているものが多いようです。このため、最近の画像作成ツールはほとんどのものが HDF データ対応となってきました。HDF 形式に対応しているツールのリストとしては

<http://hdf.ncsa.uiuc.edu/tools.html>

を参考にするのがよいでしょう。ここには HDF 形式のデータを扱うことのできるソフトウェアが数多く挙げられています。しかし、数が非常に多いので、この中から自分の目的に適ったソフトを見つけるのはなかなか難しいかもしれません。

NASA ゴダード宇宙飛行センター(GSFC)のデータセンター(DAAC)の Web サイトには、HDF に関する情報のページがあり、特に NASA の提供している衛星データを扱うのに推奨されるソフトウェアを、数を絞り込んで紹介しており、参考になります。この URL は以下の通りです。

http://daac.gsfc.nasa.gov/REFERENCE_DOCS/HDF/gdaac_hdf.html

ここでは、EORC で実際に使用した経験のあるソフトウェアについて、簡単に紹介をすることにします。

4.2 UNIX 用ソフトウェア

TSDIS Orbit Viewer

配布元: NASA/GSFC が開発。

URL: http://tsdis02.nascom.nasa.gov/tsdis/tsdis_redesign/software.html

主な特色: IDL(Interactive Data Language) をベースとしている。TRMM データ用に作られていて操作が非常に簡単。1 ファイルに対応する 1 軌道が全球マップ上に表示され、特に拡大してみたい地域を選択したり、パス沿い、パスに直角、特定高度の各断面図を表示することができる。ただし、クイックルック的。

価格: フリーソフトだが、IDL Ver.5 以上のインストールが必要。IDL の購入については例えば <http://www.adamnet.co.jp/scs/products/idl/index.html> を参照。UNIX 用の IDL の価格は約 60 万円(一般向け価格。教育機関向け価格あり)、Mac、Win 用 IDL は約 30 万円(一般向け価格。教育機関向け価格あり)だが、EORC では PC 上での使用実績はない。

JHV (Java HDF Viewer)

配布元: NCSA が開発。
URL: <http://hdf.ncsa.uiuc.edu/java-hdf-html/>
主な特色: HDF データの階層構造が表示され、データを数値で表示したり、画像として表示することが可能。TRMM データに限らず様々な HDF データに対応している。ただし、3次元データの画像表示にはメモリを大量に使うらしく、メモリの豊富な WS でないと TRMM データの表示(特に PR)は厳しい可能性がある。Java がインストールされている必要がある。また、HDF ライブラリも使用する。
価格: フリーソフト。

4.3 パソコン用ソフトウェア

Noesys Transform

対応機種: Macintosh 版、Windows 95/98/NT 版、UNIX 版がある。
配布元: 日本では複数の会社から販売されている。
URL: <http://www.fortner.com/noesys/index.html> (米国)
日本側の代理店は例えば <http://www.informatiq.com/fortnersoft/fortner2.html> など。
主な特色: HDF データの階層構造が表示され、データを数値で表示することもできるし、2次元画像として表示することもできる。TRMM データに限らず様々な HDF データに対応している。また、カット&ペーストでデータを Excel 等に貼り付けることができる。ただし、TRMM の 1 パス全部を一度に表示するにはかなりのメモリが必要。注意すべき点としては、HDF データを読み込むためのソフトであるため、1 ビットがフラグであっても全て数値で読み込むようになっている(ただし、float, integer 等のデータ形式はサポートしている)。Noesys T3D という別売りのソフトウェアを使うと3次元の画像表示ができるが、EORC では使用経験がない。また、UNIX 版については EORC での使用経験はない。
価格: PC 版は約 6 万円、UNIX 版は約 7 万円程度。

4.4 参考ソフトウェア

以下に参考として、厳密には HDF 形式に完全対応していないものの、EORC で利用したことのあるソフトウェアを紹介します。

AVS/Express Viz (Application Visualization System)

対応機種: UNIX 版と Windows NT 版がある。
配布元: 日本では複数の会社から販売されている。
URL: <http://www.avs.com> (米国)
<http://www.kgt.co.jp/kgt/avs/conso/index.html> (日本での代理店リスト)
主な特色: EORC で作成した TRMM の初期画像は主にこのソフトで作成した。HDF データ形式には対応していないため、EORC では HDF ライブラリを利用して、全ての格子点の座標値を持った不規則なメッシュデータである AVS/Express Viz のデータ・フォーマットに変換して使用した。平面画像、立体画像、鳥瞰図などが表示可能。非常に美しい画像が作成できるが、TRMM データ(特に PR)の表示には、メモリ、CPU パワーともに強力でないといかなり厳しい。PC 版については、EORC では使用経験がない。
価格: UNIX 版は約 100 万円、PC 版は約 50 万円程度(一般向け価格。教育機関は割引あり)。

GrADS (Grid Analysis and Display System)

- 対応機種:** ワークステーション版 Windows 95/NT 版、DOS 版、Linux 版、Mac 版がある。
- 配布元:** COLA/IGES (Center for Ocean-Land-Atmosphere Studies/Institute for Global Environment and Society)
- URL:** <http://grads.iges.org/grads/head.html>
- 主な特色:** Ver.1.6 β 以降で HDF データの読み書きを部分的にサポートしている。しかし、格子データでないレベル 1, 2 プロダクトの表示には向かない。また、HDF 形式のデータのうち、サイエンス・データ (SDS) の部分しか扱えない (メタデータは、読み込めない)。GrADS による HDF データの読み込みの解説は次の URL (<http://www.cdc.noaa.gov/hoop/xdlopen.shtml>) を参照するとよい。ワークステーション版以外については、EORC での HDF ファイル読み込みの使用経験はない。
- 価格:** フリーソフト。

付録 1. C サンプルプログラム

以下に、C 言語による 2A25(PR), 1B01(VIRS), 1B11(TMI), 3A25(PR)の各標準プロダクトの読み込みのためのサンプルプログラムを示します。同じものは、EORC の TRMM の Web ページ (<http://www.eorc.nasda.go.jp/TRMM>) の中にある FAQ のページでも入手が可能です。

2A25 サンプル

以下のプログラムは HDF フォーマットの 2A25 プロダクトのメタデータ(ヘッダデータ)から主要部分を抜き出して書き出した上で、すべてについて時刻・geolocation (位置情報(緯度経度))・降雨強度データを標準出力としてテキストで書き出すものである。

```
/******
```

```
2A25 data check                      98/06/24 Y.Suzuki@restec
```

```
Listing 2A25 metadata, latitude, longitude, scan time,
and rain(mm/hr)
```

USAGE:

```
c_2a25rd '2A25_File_Name'
```

This program is FREEWARE, so there is NO SUPPORT.

```
*****/
```

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#include <IO.h>
#include <IO_PR.h>
#include <IO_INTR_PR.h>
```

TSDIS ツールキット一般のおまじない

PR データ用 TSDIS ツールキットを使うためのおまじない

```
#define N100 100
```

```
int main(int argc, char *argv[])
{
```

```
/* ----- Variables related to toolkit ----- */
```

以下の構造体は IO.h, IO_PR.h で定義

```
IO_HANDLE          granuleHandle2A25;
```

2A25 データを読み出す構造体

```
L2A_25_SWATHDATA   read_L2A25_data ;
```

スキャン毎のータを読み出す構造体

```
DATE_STR beginDate;
TIME_STR beginTime;
DATE_STR endDate;
TIME_STR endTime;
```

データ取得開始・終了日付および時刻を読み出す構造体

変数の宣言

```
char granuleID_L2A25[N100]; /* グラニュール ID(ファイル名) */
int dataType_L2A25;
char filemode_read, filemode_write;
int i, j, status, iScan, numberOfScan, orbitSize;
char granuleID[N100], algorithmID[N100],
algorithmVersion[N100], toolkitVersion[N100];
```

コマンドラインの引数に関する Usage の設定

```
if( argc != 2 ) {
    fprintf( stderr, " USAGE : "
        "c_2a25rd '2A25_Input_File_Name'\n" );
    return ;
}
```

ファイル名の入力

```
strcpy(granuleID_L2A25,argv[1]);
```

HDF ファイルオープン

```
/* open file */
status = TKopen(granuleID_L2A25, TK_L2A_25,
    TK_READ_ONLY, &granuleHandle2A25);
if(status != TK_SUCCESS) {
    printf( "*** ERROR -> STOP ----- "
        "Failed to open 2A25 input file: %s\n",granuleID_L2A25);
}
```

```
/* ----- check metadata ----- */
```

メタデータの出カルーチン

```
status = TKreadMetadataInt(&granuleHandle2A25,
    TK_BEGIN_DATE, &beginDate);
status = TKreadMetadataInt(&granuleHandle2A25,
    TK_BEGIN_TIME, &beginTime);
status = TKreadMetadataInt(&granuleHandle2A25,
    TK_END_DATE, &endDate);
status = TKreadMetadataInt(&granuleHandle2A25,
    TK_END_TIME, &endTime);
```

メタデータからデータの開始時刻と終了時刻の読み出し

```
int TKreadMetadataInt(IO_HANDLE *granuleHandle, int parameter, void *value);
```

parameter: 読み込みたいパラメータ名
value: パラメータに相当する値。この例の場合日付が構造体で与えられる

```

printf( " beginDate          = %d/%d/%d %n",
        beginDate.tkyear, beginDate.tkmonth, beginDate.tkday );
printf( " beginTime          = %d:%d:%d%N",
        beginTime.tkhour, beginTime.tkminute, beginTime.tksecond );
printf( " endDate            = %d/%d/%d%N",
        endDate.tkyear, endDate.tkmonth, endDate.tkday );
printf( " endTime            = %d:%d:%d%N",
        endTime.tkhour, endTime.tkminute, endTime.tksecond );

```

日付・時刻の詳細
beginDate.tkyear 年
beginDate.tkmonth 月
beginDate.tkday 日
beginTime.tkhour 時
beginTime.tkminute 分
beginTime.tksecond 秒

```

status = TKreadMetadataInt(&granuleHandle2A25,
                           TK_ORBIT_SIZE, &orbitSize);

```

1パス内のスキャン数の読み出し

```

printf( " orbitSize          = %d%N", orbitSize );
numberOfScan = orbitSize ;

```

```

status = TKreadMetadataChar(&granuleHandle2A25,
                            TK_GRANULE_ID, granuleID);

```

グラニューール ID (ファイル名) の読み出し

```

status = TKreadMetadataChar(&granuleHandle2A25,
                            TK_ALGORITHM_ID, algorithmID);

```

データ作成アルゴリズム名の読み出し。この場合 2A25。

```

status = TKreadMetadataChar(&granuleHandle2A25,
                            TK_ALGORITHM_VERSION, algorithmVersion);

```

データ作成アルゴリズムナンバーの読み出し

```

status = TKreadMetadataChar(&granuleHandle2A25,
                            TK_TOOLKIT_VERSION, toolkitVersion);

```

データ作成に使用した TSDIS ツールキットのバージョンの読み出し

```

printf( " granuleID          = %s%N", granuleID );
printf( " algorithmID        = %s%N", algorithmID );
printf( " algorithmVersion = %s%N", algorithmVersion );
printf( " toolkitVersion       = %s%N", toolkitVersion );

```

```

/* read scan by scan */

```

データ読み込みループ開始

```

for(iScan=1; iScan<=numberOfScan; iScan++) { /* scan loop */

```

1パスデータのスキャン数だけループを回す

```

status = TKreadScan(&granuleHandle2A25,&read_L2A25_data);

```

```

int TKreadScan(IO_HANDLE *granuleHandle, void *swathData);

```

swathData: 1 スキャン分のデータの構造体。この構造体の中に geolocation、rain などのデータ配列が含まれる。

```

if( status != TK_SUCCESS ) printf( " Read Scan Error%N");

```

TSDIS ツールキットの関数は成功すると TK_SUCCESS、失敗すると TK_FAIL を返す。

```

else {
    for(i=0; i<49; i++) {

```

/* angle loop */

```

printf( "Scan : %5d/%5d  ScanTime : %9.3f  Angle : %2d  "
        "Lat,Lon : %9.3f,%9.3f \n",
        iScan, numberOfScan,
        read_L2A25_data.scanTime, i+1,
        /*   UTC seconds of the day   */
        read_L2A25_data.geolocation[i][0],
        /*   Latitude (deg) */
        read_L2A25_data.geolocation[i][1] );
        /*   Longitude (deg) */

for(j=0;j<80;j++) printf( " %8.1f",    /*   vertical loop */
        read_L2A25_data.rain[i][j] );
        /*   Rain (mm/hr) */

printf( "\n" );

    }

}

/*   close TRMM data   */
status = TKclose(&granuleHandle2A25);

return status;
}

```

1B01 サンプル

```
/******
```

```
1B01 VIRS data check          98/06/24   Y.Suzuki@restec
```

Listing 1B01 metadata information, latitude, longitude,
scan time and radiance print.

USAGE:

```
c_1b01rd '1B01_File_Name'
```

This program is FREEWARE, so there is NO SUPPORT.

```
*****/
```

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <IO.h>
#include <IO_VIRS.h>
#include <IO_INTR_VIRS.h>
```

```
#define N100    100
```

```
int main(int argc, char *argv[])
```

```
{
```

```
/* ----- Variables related to toolkit ----- */
```

```
    L1B_01_SWATHDATA    read_L1B01_data ;
    IO_HANDLE            granuleHandle1B01;
    DATE_STR    beginDate;
    TIME_STR    beginTime;
    DATE_STR    endDate;
```

```
    TIME_STR    endTime;
```

```
    char    granuleID_L1B01[N100];
    int     dataType_L1B01;
    char    filemode_read, filemode_write;
    int     i, j, status, iScan, numberOfScan, orbitSize ;
    char    granuleID[N100], algorithmID[N100],
            algorithmVersion[N100], toolkitVersion[N100] ;
```

```
if( argc != 2 ) {
    fprintf( stderr, " USAGE : "
            "c_1b01rd '1B01_Input_File_Name'\n" );
    return ;
}
```

```
strcpy(granuleID_L1B01,argv[1]);
```

```
                                /* open file */
status = TKopen(granuleID_L1B01, TK_L1B_01, TK_READ_ONLY,
                &granuleHandle1B01);

if(status != TK_SUCCESS) {
    printf( "*** ERROR -> STOP ---- "
            "Failed to open 1B01 input file: %s\n", granuleID_L1B01);
}
```

```
/* ----- check metadata ----- */
```

```
status = TKreadMetadataInt(&granuleHandle1B01,
                           TK_BEGIN_DATE, &beginDate);

status = TKreadMetadataInt(&granuleHandle1B01,
                           TK_BEGIN_TIME, &beginTime);

status = TKreadMetadataInt(&granuleHandle1B01,
                           TK_END_DATE, &endDate);

status = TKreadMetadataInt(&granuleHandle1B01,
                           TK_END_TIME, &endTime);
```

```

printf( " beginDate          = %d/%d/%d %n",
        beginDate.tkyear, beginDate.tkmonth, beginDate.tkday );
printf( " beginTime          = %d:%d:%d%N",
        beginTime.tkhour, beginTime.tkminute, beginTime.tksecond );
printf( " endDate           = %d/%d/%d%N",
        endDate.tkyear, endDate.tkmonth, endDate.tkday );
printf( " endTime           = %d:%d:%d%N",
        endTime.tkhour, endTime.tkminute, endTime.tksecond );

status = TKreadMetadataInt(&granuleHandle1B01,
                           TK_ORBIT_SIZE, &orbitSize);
printf( " numberOfScan      = %d%N", orbitSize );
numberOfScan = orbitSize ;

status = TKreadMetadataChar(&granuleHandle1B01,
                           TK_GRANULE_ID, granuleID);
status = TKreadMetadataChar(&granuleHandle1B01,
                           TK_ALGORITHM_ID, algorithmID);
status = TKreadMetadataChar(&granuleHandle1B01,
                           TK_ALGORITHM_VERSION, algorithmVersion);
status = TKreadMetadataChar(&granuleHandle1B01,
                           TK_TOOLKIT_VERSION, toolkitVersion);

printf( " granuleID         = %s%N", granuleID );
printf( " algorithmID        = %s%N", algorithmID );
printf( " algorithmVersion = %s%N", algorithmVersion );
printf( " toolkitVersion     = %s%N", toolkitVersion );

/* read scan by scan */

for(iScan=1; iScan<=numberOfScan; iScan++) { /* scan loop */

    status = TKreadScan(&granuleHandle1B01,&read_L1B01_data);

```

```

if( status != TK_SUCCESS ) printf( " Read Scan Error%N");
else {
    for(i=0; i<261; i++) { /* angle loop */

        printf( "Scan : %5d/%5d ScanTime : %9.3f IFOV : %3d "
                "Lat,Lon : %9.3f,%9.3f %N",
                iScan, numberOfScan,
                read_L1B01_data.scanTime, i+1,
                /* UTC seconds of the day */
                read_L1B01_data.geolocation[i][0],
                /* Latitude (deg) */
                read_L1B01_data.geolocation[i][1] );
                /* Longitude (deg) */

        for(j=0;j<5;j++) printf( " %9.5f", /* channel loop */
                                read_L1B01_data.channels[i][j] );
        /* radiance (mW cm^-2 um^-1 sr^-1) multiplied by a scale factor
           scale factors are 500, 1000, 100000, 10000, and 10000 */

        printf( "%N" );

    }
}

/* close TRMM data */
status = TKclose(&granuleHandle1B01);

return status;
}

```

1B11 サンプル

```
/******
```

```
1B11 TMI data check          98/06/24   Y.Suzuki@restec
```

Listing 1B11 metadata information, latitude, longitude,
scan time and Brightness temperature (K)

USAGE:

```
c_1b11rd '1B11_File_Name'
```

CALLING SEQUENCE:

(I) int argc : (number of command line strings)

(I) char *argv[] : see the note below

NOTE:

argv[0]: program name

argv[1]: input file name from 1B11

This program is FREEWARE, so there is NO SUPPORT.

```
*****/
```

```
#include <math.h>
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
#include <IO.h>
```

```
#include <IO_TMI.h>
```

```
#include <IO_INTR_TMI.h>
```

```
#define N100 100
```

```
int main(int argc, char *argv[])
```

```
{
```

```
/* ----- Variables related to toolkit ----- */
```

```
L1B_11_SWATHDATA read_L1B11_data ;
```

```
IO_HANDLE granuleHandle1B11;
```

```
DATE_STR beginDate;
```

```
TIME_STR beginTime;
```

```
DATE_STR endDate;
```

```
TIME_STR endTime;
```

```
/* ----- Other Variables ----- */
```

```
char granuleID_L1B11[N100];
```

```
int dataType_L1B11;
```

```
char filemode_read, filemode_write;
```

```
int i, j, il, status, iScan, numberOfScan, orbitSize ;
```

```
char granuleID[N100], algorithmID[N100],  
algorithmVersion[N100], toolkitVersion[N100] ;
```

```
/* ----- Argument check ----- */
```

```
if( argc != 2 ) {
```

```
fprintf( stderr, " USAGE : "
```

```
"c_1b11rd '1B11_Input_File_Name'\n" );
```

```
return ;
```

```
}
```

```
/* ----- Input file ----- */
```

```
strcpy(granuleID_L1B11,argv[1]);
```

```
dataType_L1B11 = TK_L1B_11;
```

```
filemode_read = TK_READ_ONLY;
```

```
filemode_write = TK_NEW_FILE;
```

```
status = TKopen(granuleID_L1B11, dataType_L1B11, filemode_read,  
                &granuleHandle1B11);
```

```
if(status != TK_SUCCESS) {
```

```
printf( "*** ERROR -> STOP ---- "
```

```
"Failed to open 1B11 input file: %s\n",granuleID_L1B11);
```

```

}

/* ----- check metadata ----- */
status = TKreadMetadataInt(&granuleHandle1B11,
                           TK_BEGIN_DATE, &beginDate);
status = TKreadMetadataInt(&granuleHandle1B11,
                           TK_BEGIN_TIME, &beginTime);
status = TKreadMetadataInt(&granuleHandle1B11,
                           TK_END_DATE, &endDate);
status = TKreadMetadataInt(&granuleHandle1B11,
                           TK_END_TIME, &endTime);

printf( " beginDate      = %d/%d/%d %n",
        beginDate.tkyear, beginDate.tkmonth, beginDate.tkday );
printf( " beginTime      = %d:%d:%d%N",
        beginTime.tkhour, beginTime.tkminute, beginTime.tksecond );
printf( " endDate        = %d/%d/%d%N",
        endDate.tkyear, endDate.tkmonth, endDate.tkday );
printf( " endTime        = %d:%d:%d%N",
        endTime.tkhour, endTime.tkminute, endTime.tksecond );

status = TKreadMetadataInt(&granuleHandle1B11,
                           TK_ORBIT_SIZE, &orbitSize);
printf( " numberOfScan    = %d%N", orbitSize );
numberOfScan = orbitSize ;

status = TKreadMetadataChar(&granuleHandle1B11,
                           TK_GRANULE_ID, granuleID);
status = TKreadMetadataChar(&granuleHandle1B11,
                           TK_ALGORITHM_ID, algorithmID);
status = TKreadMetadataChar(&granuleHandle1B11,
                           TK_ALGORITHM_VERSION, algorithmVersion);
status = TKreadMetadataChar(&granuleHandle1B11,
                           TK_TOOLKIT_VERSION, toolkitVersion);

```

```

printf( " granuleID      = %s%N", granuleID );
printf( " algorithmID     = %s%N", algorithmID );
printf( " algorithmVersion = %s%N", algorithmVersion );
printf( " toolkitVersion    = %s%N", toolkitVersion );

/* read scan by scan */

for(iScan=1; iScan<=numberOfScan; iScan++) { /* scan loop */

    status = TKreadScan(&granuleHandle1B11,&read_L1B11_data);

    if( status != TK_SUCCESS ) printf( " Read Scan Error%N");
    else {
        for(i=0; i<208; i++) { /* angle loop */

            printf( "Scan : %5d/%5d  ScanTime : %4d/%2d/%2d %2d:%2d:%2d"
                    " IFOV : %2d "
                    "Lat,Lon : %9.3f,%9.3f %N",
                    iScan, numberOfScan,
                    read_L1B11_data.scanTime.year,
                    read_L1B11_data.scanTime.month,
                    read_L1B11_data.scanTime.dayOfMonth,
                    read_L1B11_data.scanTime.hour,
                    read_L1B11_data.scanTime.minute,
                    read_L1B11_data.scanTime.second,
                    i+1,
                    read_L1B11_data.geolocation[i][0],
                    /* Latitude (deg) */
                    read_L1B11_data.geolocation[i][1] );
                    /* Longitude (deg) */

        }

        /* lowResCh
           Ch 1 : 10GHz Vertical
           Ch 2 : 10GHz Horizontal
        */
    }
}

```

Ch 3 : 19GHz Vertical
 Ch 4 : 19GHz Horizontal
 Ch 5 : 21GHz Vertical
 Ch 6 : 37GHz Vertical
 Ch 7 : 37GHz Horizontal

highResCh

Ch 1 : 85GHz Vertical
 Ch 2 : 85GHz Horizontal */

```
if( i%2 == 0 ) {
    il = i/2 ;
    for(j=0;j<7;j++) printf( " %8.3f",
        read_L1B11_data.lowResCh[il][j] );
    for(j=0;j<2;j++) printf( " %8.3f",
        read_L1B11_data.highResCh[i][j] );
}
else {
    for(j=0;j<63;j++) printf( " " );
    for(j=0;j<2;j++) printf( " %8.3f",
        read_L1B11_data.highResCh[i][j] );
}
```

printf("%n");

```
}
}
}
```

```
/* close TRMM data */
status = TKclose(&granuleHandle1B11);

return status;
}
```

3A25 サンプル

/******

3A25 data check

Listing 3A25 metadata and rainfall (mm/hr) for grid1

Usage: c_3a25rd '3A25_File_Name'

98/06/24 Original program by M.Kachi@EORC

Modified to C-program by S.Shimizu@EORC

This program is FREEWARE, so there is NO SUPPORT

*****/

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
```

```
#include <IO.h>
#include <IO_PR.h>
#include <IO_INTR_PR.h>
```

```
#define IDIM 16
#define JDIM 72
#define KDIM 6
#define N100 100
```

```
int main(int argc, char *argv[]){
```

/* ----- Variables related to toolkit ----- */

```
IO_HANDLE    granuleHandle3A25;
L3A_25_GRID  L3A25Grid;
```

```

    DATE_STR    beginDate;
    TIME_STR    beginTime;
    DATE_STR    endDate;
    TIME_STR    endTime;

/* Variables */

    int status;
    float mean[IDIM][JDIM][KDIM];
    char granuleID_L3A25[N100];

    if(argc != 2){
        fprintf(stderr,"USAGE: "
                    "c_3a25rd '2A25_Input_File_Name'\n");
        return;
    }

    strcpy(granuleID_L3A25, argv[1]);

/* ----- Open input file ----- */

    status = TKopen(granuleID_L3A25, TK_L3A_25, TK_READ_ONLY,
&granuleHandle3A25);

    if(status != TK_SUCCESS){
        printf("*** ERROR -> stop ----"
                "Failed to open 3A25 input file:%s\n",
granuleID_L3A25);
        return;
    }

/* ----- Check metadata ----- */

```

```

    status = TKreadMetadataInt(&granuleHandle3A25,
                                TK_BEGIN_DATE, &beginDate);
    status = TKreadMetadataInt(&granuleHandle3A25,
                                TK_BEGIN_TIME, &beginTime);
    status = TKreadMetadataInt(&granuleHandle3A25,
                                TK_END_DATE, &endDate);
    status = TKreadMetadataInt(&granuleHandle3A25,
                                TK_END_TIME, &endTime);

    printf( " beginDate      = %d/%d/%d \n",
                                beginDate.tkyear, beginDate.tkmonth, beginDate.tkday );
    printf( " beginTime      = %d:%d:%d\n",
                                beginTime.tkhour, beginTime.tkminute, beginTime.tksecond );
    printf( " endDate        = %d/%d/%d\n",
                                endDate.tkyear, endDate.tkmonth, endDate.tkday );
    printf( " endTime          = %d:%d:%d\n",
                                endTime.tkhour, endTime.tkminute, endTime.tksecond );

/* ----- Read grid data ----- */
    status = TKreadGrid(&granuleHandle3A25, &L3A25Grid);

    if( status != TK_SUCCESS ) printf(" TK read error!\n");

/* ----- Write hourly rainfall ----- */
    printf("Lat=16 Lon=72 lev=1 rain=%f\n",L3A25Grid.grid1.rainMean1[0][71][15]);

/* ----- Close ----- */
    status = TKclose(&granuleHandle3A25);

    return status;

}

```

付録 2. F77 サンプルプログラム

以下に、フォートランによる 2A25(PR), 1B01(VIRS), 1B11(TMI), 3A25(PR)の各標準プロダクトの読み込みのためのサンプルプログラムを示します。同じものは、EORC の TRMM の Web ページ (<http://www.eorc.nasda.go.jp/TRMM>) の中にある FAQ のページでも入手が可能です。

2A25 サンプル

以下のプログラムは HDF フォーマットの 2A25 プロダクトのメタデータ(ヘッダデータ)から主要部分を抜き出して書き出した上で、すべてについて時刻・geolocation(位置情報(緯度経度))・降雨強度データを標準出力としてテキストで書き出すものである。

```
*****
* 2A25 check                      98/06/24   Y.Suzuki@restec
*
*   Listing 2A25 metadata information, latitude, longitude,
*   scan time and rain(mm/hr).
*
* USAGE:
*       f_2a25rd '2A25_File_Name'
*
*   This program is FREEWARE, so there is NO SUPPORT.
*****
```

```
#include "TKfortranDeclare.h"
#include "IO.h"
#include "IO_PR.h"
```

フォートランプログラム用のおまじない

TSDIS ツールキット一般のおまじない

PR データ用 TSDIS ツールキットを使うためのおまじない

c Variables related to toolkit

以下の構造体は IO.h, IO_PR.h で定義

record /WRAPPER_HANDLE/ granuleHandle2A25

2A25 データを読み出す構造体

record /L2A_25_SWATHDATA/ read_L2A25_data

スキャン毎データを読み出す構造体

```

record /DATE_STR/ beginDate
record /TIME_STR/ beginTime
record /DATE_STR/ endDate
record /TIME_STR/ endTime

```

データ取得開始・終了日付および時刻を読み出す構造体

変数の宣言

```

c Other variables
integer status
character*100 argv(1)
character*50 granuleID, algorithmID,
* algorithmVersion, toolkitVersion

```

```

c Define input and output file names
marg = iargc()
if( marg .ne. 1 ) then
    write(6,*) ' USAGE : f_2a25rd 2A25_FileName '
    stop
end if
call getarg( 1, argv(1) )

```

コマンドラインの引数に関する Usage の設定

ファイル名の入力

```

c Open input file
write(6,*) argv(1)
status = TKopen( argv(1), TK_L2A_25, TK_READ_ONLY,
* granuleHandle2A25)
if(status .ne. TK_SUCCESS) then
    write(6,*) ' TRMM file open error'
    stop
end if

```

HDF ファイルオープン

```

c check metadata

```

メタデータの出カルーチン

```

status = TKreadMetadataInt(granuleHandle2A25,
* TK_BEGIN_DATE, beginDate)
status = TKreadMetadataInt(granuleHandle2A25,
* TK_BEGIN_TIME, beginTime)
status = TKreadMetadataInt(granuleHandle2A25,
* TK_END_DATE, endDate)
status = TKreadMetadataInt(granuleHandle2A25,
* TK_END_TIME, endTime)

```

メタデータからデータの開始時刻と終了時刻の読み出し

TKreadMetadataInt(*granuleHandle*, *parameter*, *value*)

granuleHandle: WERAPPER_HANDLE で宣言した入出力構造体名
parameter: 読み込みたいメタデータのパラメータ名
value: パラメータに相当する値。この例では日付が構造体の形で与えられる

```

write(6,'(1x,a,i4,a,i2,a,i2)')
*      ' beginDate      = ',beginDate.tkyear,'/',
*      beginDate.tkmonth,'/', beginDate.tkday
write(6,'(1x,a,i4,a,i2,a,i2)')
*      ' beginTime      = ',beginTime.tkhour,':',
*      beginTime.tkminute,':', beginTime.tksecond
write(6,'(1x,a,i4,a,i2,a,i2)')
*      ' endDate       = ',endDate.tkyear,'/',
v *      endDate.tkmonth,'/', endDate.tkday
write(6,'(1x,a,i4,a,i2,a,i2)')
*      ' endTime       = ',endTime.tkhour,':',
*      endTime.tkminute,':', endTime.tksecond

```

日付・時刻の詳細
beginDate.tkyear 年
beginDate.tkmonth 月
beginDate.tkday 日
beginTime.tkhour 時
beginTime.tkminute 分
beginTime.tksecond 秒

```

status = TKreadMetadataInt(granuleHandle2A25,
*      TK_ORBIT_SIZE, iorbitSize)
write(6,*) ' orbitSize      = ', iorbitSize
numberOfScan = iorbitSize

```

1 パス内のスキャン数の読み出し

```

status = TKreadMetadataChar(granuleHandle2A25,
*      TK_GRANULE_ID, granuleID)

```

グラニューール ID(ファイル名)の読み出し

```

status = TKreadMetadataChar(granuleHandle2A25,
*      TK_ALGORITHM_ID, algorithmID)

```

データ作成アルゴリズム名の読み出し。この場合 2A25。

```

status = TKreadMetadataChar(granuleHandle2A25,
*      TK_ALGORITHM_VERSION, algorithmVersion)

```

データ作成アルゴリズム
ナンバーの読み出し

```

status = TKreadMetadataChar(granuleHandle2A25,
*      TK_TOOLKIT_VERSION, toolkitVersion)

```

データ作成に使用した
TSDIS ツールキットのバージョンの読み出し

```

write(6,*) ' granuleID      = ', granuleID
write(6,*) ' algorithmID    = ', algorithmID
write(6,*) ' algorithmVersion = ', algorithmVersion
write(6,*) ' toolkitVersion  = ', toolkitVersion

```

c read scan by scan

データ読み込みループ開始

c scan loop

```
do 10 iScan=1,numberOfScan
```

1 パスデータのスキャン数だけループを回す

```
status = TKreadScan(granuleHandle2A25,read_L2A25_data)
```

TKreadScan(granuleHandle, swathData)

granuleHandle: WRAPPER HANDLE で宣言した入出力構造体名
swathData: 1 スキャン分のデータの構造体。この構造体の中に geolocation、rain などのデータ配列が含まれる。

```

if( status .ne. TK_SUCCESS ) then
  write(6,*) ' Read Scan Error'
else

```

TSDIS ツールキットの関数は成功すると TK_SUCCESS、失敗すると TK_FAIL を返す。

```

c      angle loop
      do 20 i=1,49

        write(6,600) iScan, numberOfScan,
c      UTC seconds of the day
        *      read_L2A25_data.scanTime, i,
c      Latitude (deg)
        *      read_L2A25_data.geolocation(1,i),
c      Longitude (deg)
        *      read_L2A25_data.geolocation(2,i)
600    format( ' scan = ', i5, '/', i5, ' ScanTime : ', f9.3,
        *      ' Angle : ', i2, ' Lat,Lon : ', f9.3, ',', f9.3)

c      Rain (mm/hr) from top to bottom
      write(6,610) (read_L2A25_data.rain(j,i),j=1,80)
610    format( 80(1x,f8.1) )

      20    continue
      end if
      10 continue

c      close TRMM data
      status = TKclose(granuleHandle2A25)

      stop
      end

```

1B01 サンプル

```
*****
* 1B01 VIRS data check
* FORTRAN version          98/06/24  Y.Suzuki@restec
*
*   Listing 1B01 metadata information, latitude, longitude,
*   scan time and radiance.
*
*USAGE:
*   f_1b01rd '1B01_File_Name'
*
*   This program is FREEWARE, so there is NO SUPPORT.
*****

#include "TKfortranDeclare.h"
#include "IO.h"
#include "IO_VIRS.h"

c   Variables related to toolkit
    record /WRAPPER_HANDLE/   granuleHandle1B01
    record /L1B_01_SWATHDATA/ read_L1B01_data
    record /DATE_STR/   beginDate
    record /TIME_STR/   beginTime
    record /DATE_STR/   endDate
    record /TIME_STR/   endTime

c   Other variables
    integer status
    character*100 argv(1)
    character*50 granuleID, algorithmID,
    *           algorithmVersion, toolkitVersion
```

```
c   Define input and output file names
    marg = iargc()
    if( marg .ne. 1 ) then
        write(6,*) ' USAGE : f_1b01rd 1B01_FileName '
        stop
    end if
    call getarg( 1, argv(1) )

c   Open input file
    write(6,*) argv(1)
    status = TKopen( argv(1), TK_L1B_01, TK_READ_ONLY,
    *               granuleHandle1B01)
    if(status .ne. TK_SUCCESS) then
        write(6,*) ' TRMM file open error'
        stop
    end if

c   check metadata

    status = TKreadMetadataInt(granuleHandle1B01,
    *               TK_BEGIN_DATE, beginDate)
    status = TKreadMetadataInt(granuleHandle1B01,
    *               TK_BEGIN_TIME, beginTime)
    status = TKreadMetadataInt(granuleHandle1B01,
    *               TK_END_DATE, endDate)
    status = TKreadMetadataInt(granuleHandle1B01,
    *               TK_END_TIME, endTime)

    write(6, '(1x,a,i4,a,i2,a,i2)')
    *           ' beginDate          = ', beginDate.tkyear, '/',
    *           beginDate.tkmonth, '/', beginDate.tkday
    write(6, '(1x,a,i4,a,i2,a,i2)')
```

```

*      ' beginTime      = ',beginTime.tkhour,':',
*      beginTime.tkminute,':', beginTime.tksecond
write(6,('1x,a,i4,a,i2,a,i2'))
*      ' endDate       = ',endDate.tkyear, '/',
*      endDate.tkmonth, '/', endDate.tkday
write(6,('1x,a,i4,a,i2,a,i2'))
*      ' endTime       = ',endTime.tkhour,':',
*      endTime.tkminute,':', endTime.tksecond

status = TKreadMetadataInt(granuleHandle1B01,
*      TK_ORBIT_SIZE, iorbitSize)
write(6,*) ' orbitSize      = ', iorbitSize
numberOfScan = iorbitSize

status = TKreadMetadataChar(granuleHandle1B01,
*      TK_GRANULE_ID, granuleID)
status = TKreadMetadataChar(granuleHandle1B01,
*      TK_ALGORITHM_ID, algorithmID)
status = TKreadMetadataChar(granuleHandle1B01,
*      TK_ALGORITHM_VERSION, algorithmVersion)
status = TKreadMetadataChar(granuleHandle1B01,
*      TK_TOOLKIT_VERSION, toolkitVersion)
write(6,*) ' granuleID      = ', granuleID
write(6,*) ' algorithmID    = ', algorithmID
write(6,*) ' algorithmVersion = ', algorithmVersion
write(6,*) ' toolkitVersion  = ', toolkitVersion

c  read scan by scan  */

c  scan loop
do 10 iScan=1,numberOfScan

    status = TKreadScan(granuleHandle1B01,read_L1B01_data)

```

```

if( status .ne. TK_SUCCESS ) then
    write(6,*) ' Read Scan Error'
else

c      angle loop
do 20 i=1,261

    write(6,600) iScan, numberOfScan,

c      UTC seconds of the day
*      read_L1B01_data.scanTime, i,
c      Latitude (deg)
*      read_L1B01_data.geolocation(1,i),
c      Longitude (deg)
*      read_L1B01_data.geolocation(2,i)
600  format( ' scan = ',i5,/,i5, ' ScanTime : ',f9.3,
*      ' IFOV : ',i3, ' Lat,Lon : ',f9.3,',',f9.3)

c      radiance (mW cm^-2 um^-1 sr^-1) multiplied by a scale factor
c      scale factors are 500, 1000, 100000, 10000, and 10000
write(6,610) (read_L1B01_data.channels(j,i),j=1,5)
610  format( 5(1x,f9.5) )

20  continue
end if
10  continue

c  close TRMM data
status = TKclose(granuleHandle1B01)

stop
end

```

1B11 サンプル

```

c-----
c  TMI/1B11 data                      98/06/24  Y.Suzuki@restec
c
c  TMI/1B11 data input
c
c  USAGE:
c      f_1b11rd '1B11_Input_File_Name'
c
c  This program is FREEWARE, so there is NO SUPPORT.
c-----

```

```

#include "TKfortranDeclare.h"
#include "IO.h"
#include "IO_TMI.h"

```

```

      record /WRAPPER_HANDLE/   granuleHandleRead1B11
      record /L1B_11_SWATHDATA/ read_L1B11_data
      record /DATE_STR/   beginDate
      record /TIME_STR/   beginTime
      record /DATE_STR/   endDate
      record /TIME_STR/   endTime

```

```

      integer status
      character*100 argv(2)
      character*50 granuleID, algorithmID,
      *           algorithmVersion, toolkitVersion

```

```

c  Define input file names
      marg = iargc()
      if( marg .ne. 1 ) then
          write(6,*)
      *  ' USAGE : f_1b11rd 1B11_InputFileName'

```

```

          stop
      end if
      call getarg( 1, argv(1) )

```

```

c  Open input file
      write(6,*) 'Input   File : ',argv(1)
      status = TKopen( argv(1), TK_L1B_11, TK_READ_ONLY,
      *               granuleHandleRead1B11)
      if(status .ne. TK_SUCCESS) then
          write(6,*) ' TRMM input file open error'
          stop
      end if

```

```

c  check metadata

```

```

      status = TKreadMetadataInt(granuleHandleRead1B11,
      *               TK_BEGIN_DATE, beginDate)
      status = TKreadMetadataInt(granuleHandleRead1B11,
      *               TK_BEGIN_TIME, beginTime)
      status = TKreadMetadataInt(granuleHandleRead1B11,
      *               TK_END_DATE, endDate)
      status = TKreadMetadataInt(granuleHandleRead1B11,
      *               TK_END_TIME, endTime)
      write(6,'(1x,a,i4,a,i2,a,i2)')
      *           ' beginDate           = ',beginDate.tkyear,'/',
      *           beginDate.tkmonth,'/', beginDate.tkday
      write(6,'(1x,a,i4,a,i2,a,i2)')
      *           ' beginTime           = ',beginTime.tkhour,':',
      *           beginTime.tkminute,':', beginTime.tksecond
      write(6,'(1x,a,i4,a,i2,a,i2)')
      *           ' endDate             = ',endDate.tkyear,'/',
      *           endDate.tkmonth,'/', endDate.tkday
      write(6,'(1x,a,i4,a,i2,a,i2)')
      *           ' endTime             = ',endTime.tkhour,':',

```

```

*           endTime.tkminute,':', endTime.tksecond

status = TKreadMetadataInt(granuleHandleRead1B11,
*           TK_ORBIT_SIZE, iorbitSize)
write(6,*) ' orbitSize           = ', iorbitSize
numberOfScan = iorbitSize

c  read scan by scan  */
do 10 iScan=1,numberOfScan

    status = TKreadScan(granuleHandleRead1B11,read_L1B11_data)

    if( status .ne. TK_SUCCESS ) then
        write(6,*) ' Read Scan Error'
    else
        do 20 ih=1,208
            write(6,600) iScan, numberOfScan,
*           read_L1B11_data.scanTime.year,
*           read_L1B11_data.scanTime.month,
*           read_L1B11_data.scanTime.dayOfMonth,
*           read_L1B11_data.scanTime.hour,
*           read_L1B11_data.scanTime.minute,
*           read_L1B11_data.scanTime.second,
*           ih,
*           read_L1B11_data.geolocation(1,ih),
*           read_L1B11_data.geolocation(2,ih)
600    format( ' scan : ',i5,',',i5,
*           '   ScanTime : ',i4,',',i2,',',i2,i3,',',i2,',',i2,
*           '   IFOV : ',i3,
*           '   Lat,Lon : ',f9.3,',',f9.3)

c           lowResCh
c           Ch 1 : 10GHz Vertical
c           Ch 2 : 10GHz Horizontal

```

```

c           Ch 3 : 19GHz Vertical
c           Ch 4 : 19GHz Horizontal
c           Ch 5 : 21GHz Vertical
c           Ch 6 : 37GHz Vertical
c           Ch 7 : 37GHz Horizontal

c           highResCh
c           Ch 1 : 85GHz Vertical
c           Ch 2 : 85GHz Horizontal

c           Brightness temperature (K)
            if(mod(ih,2).eq.1) then
                il = (ih-1)/2+1
                write(6,610) (read_L1B11_data.lowResCh(j,il),j=1,7),
*                (read_L1B11_data.highResCh(j,ih),j=1,2)
610            format( 9(1x,f8.3) )
            else
                write(6,620) (read_L1B11_data.highResCh(j,ih),j=1,2)
620            format( 63x,2(1x,f8.3) )
            end if

20    continue
    end if
10 continue

c  close TRMM data
    status = TKclose(granuleHandleRead1B11)

    stop
    end

```

3A25 サンプル

```

c*****
c      PR_3A25 data check
c
c      Listing 3A25 metadata and rainfall(mm/hr) for grid1
c
c      Usage: % f_3a25rd '3A25_File_Name'
c
c
c      24.JUN.1998. Programmed by M.KACHI@EORC
c      This program is FREEWARE, so there is NO SUPPORT.
c*****

      program main

#include "TKfortranDeclare.h"
#include "IO.h"
#include "IO_PR.h"

      record /L3A_25_GRID/ L3A25Grid
      record /WRAPPER_HANDLE/ granuleHandle3A25

      record /DATE_STR/   beginDate
      record /TIME_STR/   beginTime
      record /DATE_STR/   endDate
      record /TIME_STR/   endTime

c      Constants
      parameter (idim=72, jdim=16, kdim=6)

c      Variables
      integer status
      character*100 argv

```

```

c      Begin
c      Define input file name
      marg = iargc()
      if( marg .ne. 1 ) then
         write(6,*) ' USAGE: 3A25_out 3A25_InputFileName'
         stop
      end if
      call getarg( 1, argv )

c      Open input file
      write(6,*) 'Input   File : ', argv
      status = TKopen( argv, TK_L3A_25, TK_READ_ONLY,
$   granuleHandle3A25)
      if(status .ne. TK_SUCCESS) then
         write(6,*) ' TRMM input file open error'
         stop
      end if

c      Check metadata
      status = TKreadMetadataInt(granuleHandle3A25,
$   TK_BEGIN_DATE, beginDate)
      write(6,(a,i4,a,i2,a,i2))
$   ' beginDate      = ',beginDate.tkyear,'/',
$   beginDate.tkmonth,'/', beginDate.tkday

      status = TKreadMetadataInt(granuleHandle3A25,
$   TK_BEGIN_TIME, beginTime)
      write(6,(a,i4,a,i2,a,i2))
$   ' beginTime(UTC) = ',beginTime.tkhour,':',
$   beginTime.tkminute,':', beginTime.tksecond

      status = TKreadMetadataInt(granuleHandle3A25,
$   TK_END_DATE, endDate)
      write(6,(a,i4,a,i2,a,i2))

```

```

$ ' endDate      = 'endDate.tkyear, '/',
$  endDate.tkmonth, '/', endDate.tkday
status = TKreadMetadataInt(granuleHandle3A25,
$  TK_END_TIME, endTime)
write(6, '(a,i4,a,i2,a,i2)')
$ ' endTime(UTC)  = 'endTime.tkhour, ':',
$  endTime.tkminute, ':', endTime.tksecond

c      Read grid data
status = TKreadGrid(granuleHandle3A25, L3A25Grid)

if( status .ne. TK_SUCCESS ) then
  write(6, *) 'TK read error!'
  stop
endif

c      Write hourly rainfall
write(6, *) 'Lat=16 Lon=72 lev=1 rain=',
$  L3A25Grid.grid1.rainMean1(16,72,1)

c      Close
status = TKclose(granuleHandle3A25)

c      End
stop
end

```