GPM/TRMM
データ読み込みプログラムガイド
(FORTRAN 編)

2018/03/15

第三版

本書は全球降雨観測衛星(GPM)のデータを読み込むプログラム（FORTRAN）の作成方法についてまとめたものです。
本書で解説するサンプルプログラムは、GPM/TRMM はプロダクトバージョン０6、GSMaP はプロダクトバージョン０４で動作を確認しています。

目次

1. はじめに

本書は GPM/TRMM データに対して FORTRAN 言語を用いて読み込む方法について解説します。
TRMM バージョン 8 相当プロダクトは、GPM バージョン 06 とフォーマットを統一し、GPM/TRMM バージョン 06 としてリリースされました。本サンプルプログラムにた同様に読むことができます。
GPM データを読み込むには FORTRAN の他にも表 1.1 に示すような方法があります。どの方法で読み込むかについては、次頁の「読み込み方法判断フロー」を参考にして判断してください。
また、本資料で使用しているサンプルプログラムの動作を確認したＯＳの一覧を表 1.2 に示します。

表 1.1 データ読み込み方法

	データ読み込み方法	資料名	備考
1	THOR を使用する	GPM/TRMM データ読み込みプログラムガイド(THOR 編)	
2	IDL を使用する	GPM/TRMM データ読み込みプログラムガイド(IDL 編)	
3	C を使用する	GPM/TRMM データ読み込みプログラムガイド(C 言語編)	
4	FORTRAN を使用する	GPM/TRMM データ読み込みプログラムガイド(FORTRAN 編)	
5	Python を使用する	GPM/TRMM データ読み込みプログラムガイド(Python 編)	

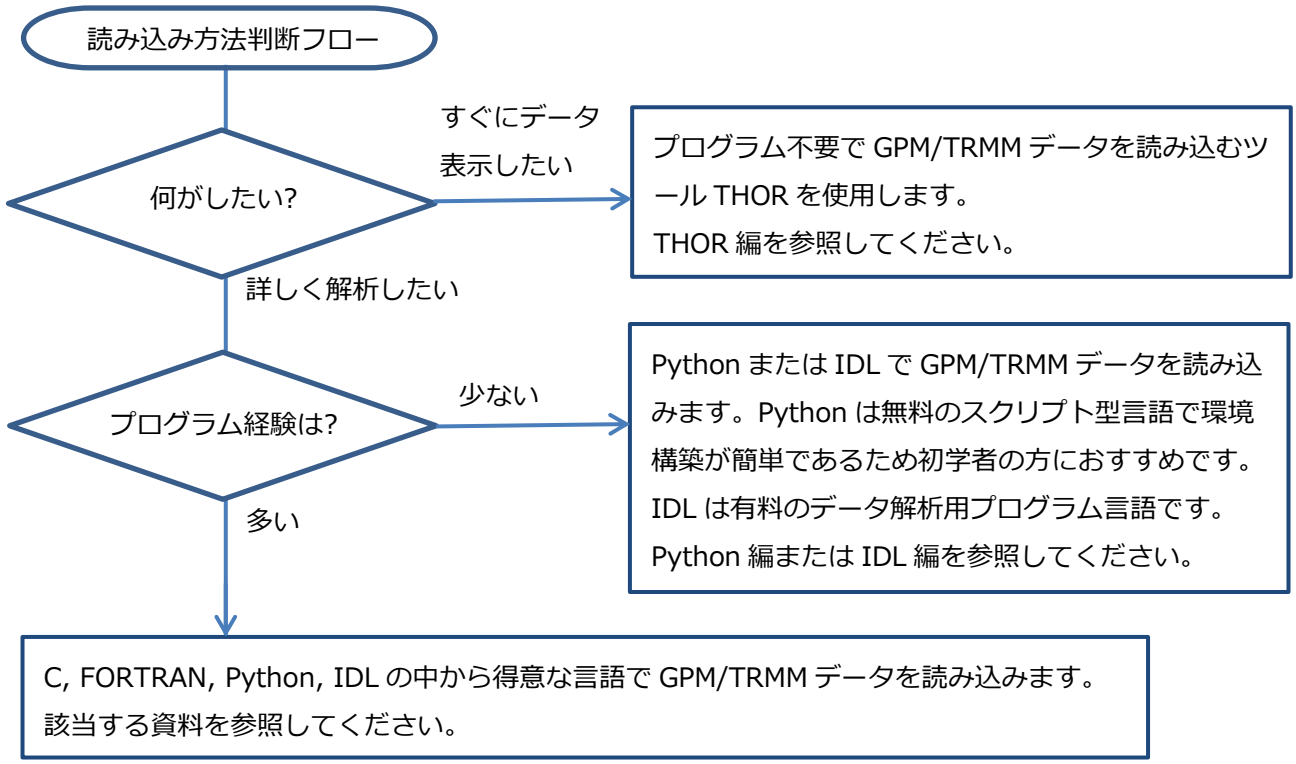


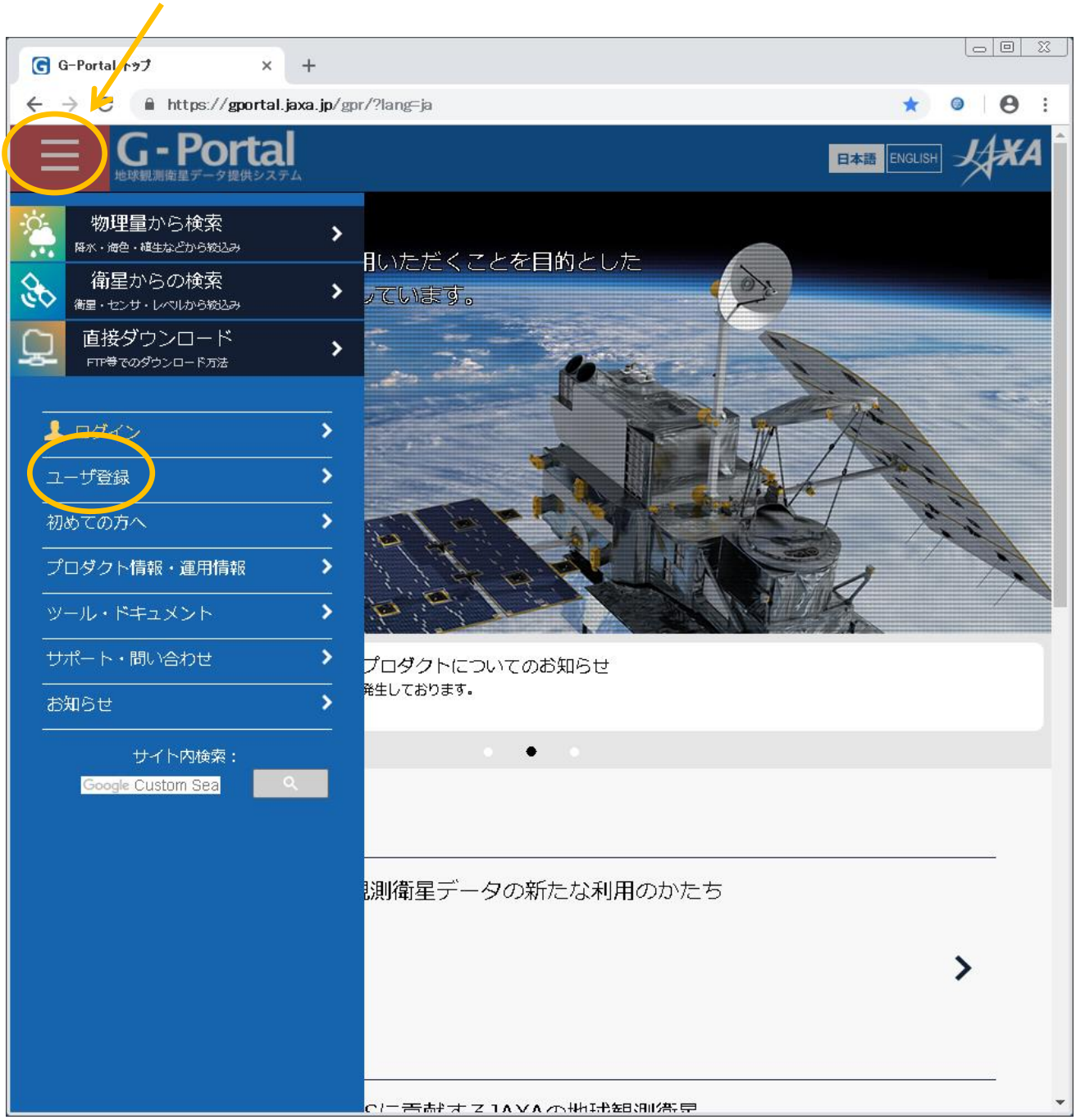
表 1.2 サンプルプログラム動作確認表

	サンプルプログラム	Linux	Windows	備考
1	C	○	—	
2	Fortran	○	—	
3	Python	○	○	
4	IDL	○	○	

2. GPM/TRMM データの入手方法

GPM/TRMM データは、G-Portal のサイト(https://www.gportal.jaxa.jp/gp/top.html)から取得することができます。 取得の際にはユーザ登録が必要になりますので、G-Portal のサイトのメニューから「ユーザ登録」を選択してユーザ登録を行ってください。

ここをクリックしてメニューを表示



規約を読み「同意して次へ」をクリックします。

ユーザ登録画面になりますので、ユーザ登録を行います。

以降の手順や、ユーザ登録後のデータ取得方法については、「GPM データ利用ハンドブック」の「5.2 データ提供サービスの使い方」を参照してください。「GPM データ利用ハンドブック」の入手方法については「3. 関連文書、サンプルプログラムの入手方法」を参照してください。

3. 関連文書、サンプルプログラムの入手方法

GPM/TRMM データの関連文書には、GPM データ利用に関する文書と、プロダクトに関する文書があります。どちらも全球降水観測計画 GPM のサイト(https://www.eorc.jaxa.jp/GPM/index.html)のトップページ ＞ 資料を読む ＞ その他 からダウンロードできます。また、本書で解説しているサンプルコードについてもこちらからダウンロードできます。

GPM データ利用に関する文書には以下のものがあります。

GPM データ利用ハンドブック

ファイル命名規約

「TRMM/GPM V06」をクリックするとプロダクトバージョン 06 の文書一覧が表示されます。Format Specification は各プロダクトのデータ仕様が記載されたドキュメントです。

本書で解説するプロダクトとプログラム、サンプルデータは以下の通りです。

表 3.1 サンプルプログラム一覧

プロダクト	サンプルプログラム	サンプルデータ
L1Ku	sample_L1_Ku_F.f90	GPMCOR_KUR_1708202148_2320_019762_1BS_DUB_05A.h5
L2DPR	sample_L2_DPR_F.f90	GPMCOR_DPR_1512282046_2218_010412_L2S_DD2_06A.h5
	sample_HDF5_L2_DPR_F.f90	
	sample_h5dump_L2_F.f90	
L3DPR	sample_L3_DPR_F.f90	GPMCOR_DPR_1407_M_L3S_D3M_06A.h5
	sample_HDF5_L3_DPR_F.f90	
	sample_h5dump_L3_F.f90	
L2GMI	sample_L2_GMI_F.f90	GPMCOR_GMI_1709252152_2324_020322_L2S_GL2_05A.h5
L3GMI	sample_L3_GMI_F.f90	GPMCOR_GMI_1707_M_L3S_GL3_05A.h5
L2CMB	sample_L2_CMB_F.f90	GPMCOR_CMB_1901082158_2331_027633_L2S_CL2_06A.h5
L3CMB	sample_L3_CMB_F.f90	GPMCOR_CMB_1812_M_L3S_CL3_06A.h5
L1PR	sample_L1_PR_F.f90	GPMTRM_PR1_1503251901_2032_098882_1BS_PU1_8b21.h5
L2PR	sample_L2_PR_F.f90	GPMTRM_PR1_1503251901_2032_098882_L2S_PU2_8b21.h5
L3PR	sample_L3_PR_F.f90	GPMTRM_KUR_1406_M_D3M_06A.h5
GSMaP	sample_GSMaP_HDF5_F.f90	GPMMRG_MAP_1709242300_H_L3S_MCH_04D.h5

4. ライブラリ・ツールのインストール

FORTRAN で GPM データを読み込むには、表 4.1 で示すように 3 種類の方法があり、方法によってはツールをインストールする必要があります。本書ではそれぞれについてプログラム作成の解説を行います。

表 4.1 GPM データ読み込み方法

	GPM データ読み込み方法	必要なライブラリ、ツール	備考
1	PPS Toolkit(TKIO)	HDF5、PPS TKIO	
2	HDF5 ライブラリ	HDF5	
3	h5dump	HDF5	

また PPS Toolkit(TKIO)を利用する場合、GPM データのプロダクトバージョンと、対応する PPS Toolkit(TKIO)バージョンの関係を表 4.2 に示します。

表 4.2 プロダクトバージョンと PPS Toolkit(TKIO)の対応バージョン

プロダクト	プロダクトバージョン	PPS ツールキットのバージョン	備考
L1Ku	05	3．80．44	
L2DPR	06	3．80．44	
L3DPR	06	3．80．44	
L2GMI	05	3．80．26	
L3GMI	05	3．80．26	
L2CMB	06	3．90	
L3CMB	06	3．90	
L1PR	05	3．80．25	
L2PR	06	3．80．25	
L3PR	06	3．80．25	
GSMaP	04	3．80．10	

注）PPS Toolkit(TKIO)は基本的には上位互換ですが、一部で正常に読み込めない場合があります。その場合は「5.9 PPS Toolkit(TKIO)のバージョンについて」を参照してください。

本書のサンプルプログラムは以下の環境で動作確認を行っています。

表 4.2 動作環境

項目	環境
計算機	Intel(R) Xeon(R) CPU ES-2665　2.4GHz
OS	Red Hat Enterprise Linux Server release 6.4
FORTRAN コンパイラ	ifort 14.0.1
HDF5	Hdf5-1.8.9
PPS TKIO	tkio-x.xx.x

4.1　HFD5 のインストール

4.1.1 ダウンロード

The HDF Group ホームページ(http://www.hdfgroup.org/)から HDF5 のソースインストール版の圧縮ファイルをダウンロードします。

※以下では hdf5-1.8.9.tar.gz をダウンロードしたものとして説明します。

4.1.2 解凍

適当な作業ディレクトリで圧縮ファイルを解凍します。 以下のコマンドで解凍できます。

```
$ tar -xzvf hdf5-1.8.9.tar.gz
```

解凍すると、hdf5-1.8.9 のようなディレクトリが作成されるので、その配下へ移動します。

```
$ cd hdf5-1.8.9
```

4.1.3 コンパイルとインストール

以下のコマンドを順番に実行して、コンパイルとインストールを行います。

--prfix=には、インストール先ディレクトリを指定します。

※この例の場合、hdf5 のバージョンは 1.8.9 なので、hdf5_1.8.9 としています。

バージョン文字部分は実際に使用するバージョンに置き換えてください。

<HDF5 の FORTRAN ライブラリを使用しない場合>

```
$ ./configure --disable-shared --prefix=/home/user1/util/hdf5_1.8.9
 --with-szlib=/home/user1/util/szip_2.1
$ make
$ make install
```

<HDF5 の FORTRAN ライブラリを使用する場合>

```
$ ./configure --disable-shared --prefix=/home/user1/util/hdf5_1.8.9
 --with-szlib=/home/user1/util/szip_2.1 --enable-fortran FC=ifort
$ make
$ make install
```

4.2　PPS Toolkit(TKIO)のインストール

PPS Toolkit(TKIO)とは、GPM の HDF5 ファイルを読み込むプログラムを作成する際に使用するライブラリです。HDF5 ライブラリを使用して読み出す場合や、h5dump を使用して読み出す場合にはインストールする必要はありません。

4.2.1 ダウンロード

以下の URL から 、自分の環境に合った圧縮ファイルをダウンロードします。

https://gpmweb2https.pps.eosdis.nasa.gov/pub/PPStoolkit/GPM/

4.2.2 解凍

適当な作業ディレクトリを作成してダウンロードしたファイルを移し、圧縮ファイルを解凍します。以下のコマンドで解凍できます。

```
$ mkdir tikio.xxx
$ mv tikio.xxx.tar.gz tikio.xxx/.
$ cd tikio.xxx
$ tar zxf tikio.xxx.tar.gz
```

4.2.3 前提条件の確認

docs ディレクトリにある tkioINSTALL.txt ファイルを参照し、ダウンロードした PPS Toolkit(TKIO)が動作する前提条件を確認します。必要なライブラリがインストールされていない場合や、バージョンが古い場合はインストールを行います。

・libxml2 ライブラリのインストール

```
必要なバージョンは docs/tkioINSTALL.txt を確認！
./configure --prefix=[インストール DIR]
make
make install
```

・zlib ライブラリのインストール

```
./configure --prefix=[インストール DIR]
make
make install
```

・jpeg ライブラリのインストール

```
./configure --prefix=[インストール DIR] --enable-shared
make
make install
make install-lib
```

・hdf4 ライブラリのインストール

```
./configure --prefix=[インストール DIR] --with-zlib=[zlib インストール DIR]
--with-jpeg=[jpeg インストール DIR] --with-szlib=[szlib インストール DIR] CC=icc F77=ifort CXX=icpc
make
make install
```

・hdf5 ライブラリのインストール

```
./configure --prefix=[インストール DIR] --enable-fortran --with-zlib=[zlib インストール DIR]
--with-szlib=[szip インストール DIR] CC=icc CXX=icpc FC=ifort
make
make install
```

4.2.4 環境設定ファイルの編集

環境変数を定義するファイルを作成します。以下に作成例を示します。自分の環境に合った環境変数を定義してください。

```
 1:unlimit
 2:setenv TKDEBUG ”-g”
 3:
 4:setenv TKIO /home/tool/tkio-x.xx.x_HDF4/tkio
 5:setenv HDF_INC /export/trmm5/tool/x86_64/HDF4.2r1/include
 6:setenv HDF_LIB /export/trmm5/tool/x86_64/HDF4.2r1/lib
 7:setenv HDF4_INC /export/trmm5/tool/x86_64/HDF4.2r1/include
 8:setenv HDF4_LIB /export/trmm5/tool/x86_64/HDF4.2r1/lib
 9:setenv HDF5_INC /export/trmm5/tool/x86_64/hdf5-1.8.9_gcc/include
10:setenv HDF5_LIB /export/trmm5/tool/x86_64/hdf5-1.8.9_gcc/lib
11:setenv CLASSPATH $TKIO/classes
12:
13:setenv SZIP_INC /home/tool/szip-2.1/include
14:setenv SZIP_LIB /home/tool/szip-2.1/lib
15:setenv xml2 /usr/include/libxml2
16:
17:setenv LD_LIBRARY_PATH ${HDF5_LIB}:${LD_LIBRARY_PATH}
18:
19:setenv CC icc
20:setenv CFLAGS ’-fPIC -mcmodel=medium’
21:setenv CXXFLAGS ’-fPIC -mcmodel=medium’
22:setenv FFLAGS ’-fPIC -mcmodel=medium’
23:setenv FC ifort
24:setenv F77 ifort
25:setenv F90 ifort
26:setenv FORTC ifort
27:
28:setenv PATH ./:/home/tool/hdf5-1.8.9/bin:$PATH
```

4.2.5 環境設定ファイルの読み込み

以下のコマンドで環境設定ファイルを読み込みます。

$ source 環境設定ファイル名

4.2.6 コンパイル

以下のコマンドでコンパイルを実行します。

$ ./INSTALL.pl compileJAVA

$ ./INSTALL.pl buildRW

$ ./INSTALL.pl compileRW

5. PPS Toolkit(TKIO)で GPM データ読み込み

PPS Toollit(TKIO)を使用した Fortran プログラムの作成方法について説明します。PPS Toollit(TKIO)を使用する場合は、予め PPS Toollit(TKIO)をインストールしておく必要があります。

また、PPS Toollit(TKIO)を使用してプログラムを作成する場合、予めアルゴリズム ID を知っておく必要があります。アルゴリズム ID とはプロダクト（データの種類）毎にある ID で、HDF5 ファイルのファイルヘッダに格納されています。PPS Viewer THOR でファイルヘッダの情報を確認することができます。

表 5.1 プロダクトとアルゴリズム ID、TKIO ヘッダファイルの対応

レベル	プロダクト	アルゴリズム ID	TKIO ヘッダファイル	備考
1	L1Ku	1BKu	TK_1BKu.h	
	L1PR	1BPR	TK_1BPR.h	
2	L2DPR	2ADPR	TK_2ADPR.h	
	L2GMI	2AGPROFGMI	TK_2AGPROFGMI.h	
	L2CMB	2BCMB	TK_2BCMB.h	
	L2PR	2APR	TK_L2APR.h	
3	L3DPR	3DPR	TK_3DPR.h	
	L3GMI	3GPROF	TK_3GPROF.h	
	L3CMB	3CMB	TK_3CMB.h	
	L3PR	3PR	TK_3PR.h	
	GSMaP	3GSMAPH	TK_3GSMAP.h	

プログラムを作成する際、読み込むデータにあわせて格納する領域を定義する必要があります。GPM/TRMM データは、スキャン、アングルビン、レンジビンという名称の次元で構成されています。このスキャン、アングルビン、レンジビンの関係については、「GPM/TRMM データ読み込みプログラムガイド(付録)」の「1.2 シーン定義」を参照してください。また、読み込むデータの構成については 「3. 関連文書、サンプルプログラムの入手方法」で示したサイトから「GPM/DPR TRMM/PR L1 プロダクトフォーマット説明書」/ 「GPM/DPR TRMM/PR L2/L3 プロダクトフォーマット説明書」をダウンロードして参照してください。

次項よりデータ読み込みプログラムの作成例を示します。

プログラムの説明は以下のように色分けしています。

赤字の解説はサンプルプログラムについて説明しています。

青字の解説は PPS Toolkit または衛星基礎知識について説明しています。

5.1 L1 データ読み込み

5.1.1 ソースプログラム

以下は L1Ku を読み込むプログラム例です。ジョブ名と、L1Ku の HDF5 ファイル名を引数として、引数として指定されたファイルから、日時情報、緯度経度情報、echoPower、noisePower というデータを読み込んでいます。

```
 1:PROGRAM SAMPLE
 2:
 3:#include "TKHEADERS.h"
 4:#include "TK_1BKu.h"
 5:#include "tkiofortdeclare.h"
 6:

 7:RECORD /TKINFO/ granuleHandle1BKu



 8:RECORD /L1BKu_NS/ L1BKu
 9:
10:    PARAMETER( NANGLE=49, NRANGE=260 )
11:
12:    INTEGER status, numOfScan
13:    INTEGER iscan, iangle, irange
14:    CHARACTER*255 jobname, file
15:    REAL*8 lat(NANGLE), lon(NANGLE)
16:    INTEGER*2 noisePower(NANGLE),
echoPower(NRANGE,NANGLE)
17:    INTEGER*2 year, msec, dayOfYear
18:    BYTE month, dayOfMon, hour, min, sec
19:    REAL*8 secOfDay
20:
21:    call getarg( 1, jobname )
22:    call getarg( 2, file )
23:
24:    ! Open the file for reading.
25:    status = TKopen( file, '1BKu', TKREAD, 'HDF5', jobname, granuleHandle1BKu, 0 )
26:    IF ( status .NE. TK_SUCCESS ) THEN
27:      status = TKmessage( granuleHandle1BKu.jobname, TKERROR,'message to print' )
28:    ENDIF
29:









30:    status = TKgetMetaInt( granuleHandle1BKu, 'SwathHeader',
'NumberScansGranule', numOfScan )
31:
```

該当するアルゴリズム ID のヘッダファイルをインクルードします(表 5.1 プロダクトとアルゴリズム ID、TKIO ヘッダファイルの対応を参照)。。L1Ku はアルゴリズム ID が"1BKu"なので"TK_1BKu.h"をインクルードします。

Fortran の場合必ずインクルードします。

granuleHandle1BKu は HDF5 ファイルの情報を格納する構造体で PPS Toolkit を使用する際に使用します。

L1BKu は HDF5 ファイルの構造体です。1 スキャン分のデータを格納します。

HDF5 ファイルのオープン
file:HDF5 ファイル名(引数で指定した文字列)
1BKu:アルゴリズム ID
TKREAD:読み込み指定
HDF5:フォーマットタイプ
jobname:ジョブ名(引数で指定した文字列)、
granuleHandle1BKu:ファイルポインタ(TKINFO 構造体を指定)
1 :ファイルの内部圧縮(1 を指定)

メタデータ読み込み（スキャン数読み出し）
granuleHandle1BKu：ファイルポインタ(TKINFO 構造体を指定)
"SwathHeader"：HDF5 ファイルにある項目名で、読み出すデータの情報が格納されています。予め項目名を「L1 プロダクトフォーマット説明書」か PPS Viewer THOR で確認しておく必要があります。
"NumberScansGranule"：スキャン数を指定
numOfScan：読み出したスキャン数を格納する変数

```
32:     do iscan=1,numOfScan
33:       status = TKreadScan( granuleHandle1BKu, L1BKu )
34:
35:       ! ScanTime Read
36:       year      = L1BKu.ScanTime.Year
37:       month     = L1BKu.ScanTime.Month
38:       dayOfMon  = L1BKu.ScanTime.DayOfMonth
39:       hour      = L1BKu.ScanTime.Hour
40:       min       = L1BKu.ScanTime.Minute
41:       sec       = L1BKu.ScanTime.Second
42:       msec      = L1BKu.ScanTime.MilliSecond
43:       dayOfYear = L1BKu.ScanTime.DayOfYear
44:       secOfDay  = L1BKu.ScanTime.SecondOfDay
45:
46:       do iangle=1,NANGLE
47:         ! Latitude and Longitude Read
48:         lat(iangle)  = L1BKu.Latitude(iangle)
49:         lon(iangle)  = L1BKu.Longitude(iangle)
50:
51:         ! noisePower Read
52:         noisePower(iangle) = L1BKu.Receiver.noisePower(iangle)
53:
54:         ! echoPower Read
55:            do irange=1,NRANGE
56:           echoPower(irange, iangle) = L1BKu.Receiver.echoPower(irange, iangle)
57:         enddo
58:
59:         ! print the value
60:         IF(iscan .eq. 3947 .and. iangle .eq. 20 ) THEN
61:           write(*,*) "scan=",iscan-1,",angle=",iangle-1
62:           write(*,*) "lat=", lat(iangle), "lon=", lon(iangle)
63:           write(*,*) "echoPower(259)=",echoPower(260,iangle)
64:           write(*,*) "noisePower=",noisePower(iangle)
65:         ENDIF
66:       enddo
67:     enddo
68:
69:     ! Close HDF5 file.
70:     status = TKclose( granuleHandle1BKu )
71:     STOP
72:END
```

スキャン数分ループ

スキャンデータ読み込み
granuleHandle1BKu：ファイルポインタ(TKINFO 構造体を指定)
L1BKu：読み出したデータを格納する領域

スキャン日時の読み込み

緯度経度情報読み込み

noisePower 読み込み

echoPower 読み込み

正しく読み込めているか確認するため、一部分(このケースは、スキャンが 3947 番目のアングル 20、レンジビン 260 のデータ）を出力しています。

HDF ファイルのクローズ
granuleHandle1BKu：ファイルポインタ(TKINFO 構造体を指定)

5.1.2 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
 1:F90= ifort
 2:MACHINE=LINUX
 3:FFLAGS=-g -CB -traceback -shared-intel -mcmodel=medium -fPIC -fpp -D$(MACHINE)
-DLANGUAGE_FORTRAN -DWRITEPIAINFO
 4:
 5:MAIN=./sample_L1_Ku_F
 6:
 7:OBJS= $(MAIN).o
 8:
 9:INC = -I$(HDF4_INC) ¥
10:     -I$(HDF5_INC) ¥
11:     -I$(TKIO)/inc/fortcode¥
12:     -I$(SZIP_INC)
13:
14:LIB = -L$(HDF4_LIB) ¥
15:     -L$(HDF5_LIB) ¥
16:     -L$(TKIO)/lib ¥
17:     -L$(SZIP_LIB)
18:
19:LIBES = -ltkcselect -ltkchdf4algs -ltkchdf4 ¥
20:     -ltkchdf5algs -ltkchdf5 -ltkctkTSDIS -ltkchelper¥
21:     -ltkc -lm -lmfhdf -ldf -lhdf5_hl -lhdf5 -ljpeg -lsz -lz -lxml2
22:
23:$(MAIN): $(OBJS)
24:     $(F90) $(FFLAGS) -o $(MAIN) $(OBJS) $(INC) $(LIB) $(LIBES)
25:
26:$(MAIN).o: $(MAIN).f90
27:     $(F90) $(FFLAGS) -c $(MAIN).f90 $(INC) $(LIB) $(LIBES)
28:clean:
29:     rm -f *.o $(MAIN)
```

5.1.1 のプログラムの名前です。

HDF5/HDF5 のインクルードディレクトリ、ライブラリディレクトリのパスです。PPS Toolkit(TKIO)を使用する場合、HDF4 も必要になるようです。

PPS Toolkit(TKIO)の Fortran のヘッダファイルがあるディレクトリのパスです

PPS Toolkit(TKIO)のライブラリがあるディレクトリのパスです

5.1.3 実行結果

5.1.1 で説明したプログラムの実行結果を示します。

第 1 引数の”cc”はジョブ名です。(任意の文字列で構いません)

第 2 引数の”/DPR/STD/…….h5”は HDF5 ファイル名です。

```
$  ./sample_L1_Ku_F "cc" "/DPR/STD/L1B/GPMCOR_KUR_sample.h5"
 scan=       3946 ,angle=         19
 lat=   64.8921890258789      lon=  -163.892868041992
 echoPower(259)= -29999
 noisePower= -11305
$
```

5.2 L2 データ読み込み

5.2.1 ソースプログラム

以下は L2DPR を読み込むプログラム例です。ジョブ名と、L2DPR の HDF5 ファイル名を引数として、引数として指定されたファイルから、日時情報、緯度経度情報、precipRateESurface というデータを読み込んでいます。

```
 1:PROGRAM SAMPLE
 2:
 3:#include "TKHEADERS.h"
 4:#include "TK_2ADPR.h"

 5:#include "tkiofortdeclare.h"
 6:

 7:RECORD /TKINFO/ granuleHandleL2DPR


 8:RECORD /L2ADPR_SWATHS/ L2ADPR
 9:
10:    PARAMETER( NANGLE=49, NRANGE=176 )
11:
12:    INTEGER status, numOfScan
13:    INTEGER iscan, iangle, irange
14:    CHARACTER*255 jobname, file
15:    REAL*8 lat(NANGLE), lon(NANGLE)
16:    REAL*4 precipEsurf(NANGLE)
17:    REAL*4 zFactorCor(NRANGE,NANGLE)
18:    INTEGER*2 year, msec, dayOfYear
19:    BYTE month, dayOfMon, hour, min, sec
20:    REAL*8 secOfDay
21:
22:    call getarg( 1, jobname )
23:    call getarg( 2, file )
24:
25:    !Open the file for reading.
26:    status = TKopen( file, '2ADPR', TKREAD, 'HDF5', jobname, granuleHandleL2DPR,
0 )
27:    IF ( status .NE. TK_SUCCESS ) THEN
28:      status = TKmessage( granuleHandleL2DPR.jobname, TKERROR,'message to print' )
29:    ENDIF
30:









31:    status = TKgetMetaInt( granuleHandleL2DPR, 'NS_SwathHeader',
'NumberScansGranule', numOfScan )
32:
```

該当するアルゴリズム ID のヘッダファイルをインクルードします(表 5.1 プロダクトとアルゴリズム ID、TKIO ヘッダファイルの対応を参照)。L2DPR はアルゴリズム ID が"2ADPR"なので"TK_2ADPR.h"をインクルードします。

Fortran の場合必ずインクルードします。

granuleHandleL2DPR は HDF5 ファイルの情報を格納する構造体で PPS Toolkit を使用する際に使用します。

L2ADPR は HDF5 ファイルの構造体です。1 スキャン分のデータを格納します。

HDF5 ファイルのオープン
file:HDF5 ファイル名(引数で指定した文字列)
2ADPR:アルゴリズム ID
TKREAD:読み込み指定
HDF5:フォーマットタイプ
jobname:ジョブ名(引数で指定した文字列)
granuleHandleL2DPR：ファイルポインタ
(TKINFO 構造体を指定)
1：ファイルの内部圧縮(1 を指定)

メタデータ読み込み（スキャン数読み出し）
granuleHandle2ADPR：ファイルポインタ(TKINFO 構造体を指定)
"NS_S granuleHandleL2DPR wathHeader"：HDF5 ファイルにある項目名で、読み出すデータの
情報が格納されています。予め項目名を「L1 プロダクトフォーマット説明書」か PPS Viewer THOR で確認しておく必要があります。
"NumberScansGranule"：スキャン数を指定
numOfScan：読み出したスキャン数を格納する変数

スキャン数分ループ

スキャンデータ読み込み
granuleHandleL2DPR：ファイルポインタ(TKINFO 構造体を指定)
L2ADPR：読み出したデータを格納する領域

スキャン日時の読み込み

緯度経度情報読み込み

precipRateESurface 読み込み

zFactorCorrected 読み込み

正しく読み込めているか確認するため、一部分（このケースは、スキャンが 3947 番目のアングル 20 のデータ）を出力しています。

HDF ファイルのクローズ
granuleHandleL2DPR：ファイルポインタ(TKINFO 構造体を指定)

```
 33:    do iscan=1,numOfScan
 34:      status = TKreadScan( granuleHandleL2DPR, L2ADPR )
 35:
 36:      ! ScanTime Read
 37:      year      = L2ADPR.NS.ScanTime.Year
 38:      month     = L2ADPR.NS.ScanTime.Month
 39:      dayOfMon  = L2ADPR.NS.ScanTime.DayOfMonth
 40:      hour      = L2ADPR.NS.ScanTime.Hour
 41:      min       = L2ADPR.NS.ScanTime.Minute
 42:      sec       = L2ADPR.NS.ScanTime.Second
 43:      msec      = L2ADPR.NS.ScanTime.MilliSecond
 44:      dayOfYear = L2ADPR.NS.ScanTime.DayOfYear
 45:      secOfDay  = L2ADPR.NS.ScanTime.SecondOfDay
 46:
 47:      do iangle=1,NANGLE
 48:
 49:        ! Latitude and Longitude Read
 50:        lat(iangle)  = L2ADPR.NS.Latitude(iangle)
 51:        lon(iangle)  = L2ADPR.NS.Longitude(iangle)
 52:
 53:        ! precipRateESurface Read
 54:        precipEsurf(iangle) = L2ADPR.NS.SLV.precipRateESurface(iangle)
 55:
 56:        do irange=1,NRANGE
 57:          zFactorCor(irange, iangle) = L2ADPR.NS.SLV.zFactorCorrected(irange,
iangle)
 58:        enddo
 59:
 60:        ! print the value
 61:        IF(iscan .eq. 3947 .and. iangle .eq. 20 ) THEN
 62:          write(*,*) "scan=",iscan-1,",angle=",iangle-1
 63:          write(*,*) "lat=", lat(iangle), "lon=", lon(iangle)
 64:          write(*,*) "precipEsurf=",precipEsurf(iangle), "(mm/h)"
 65:        ENDIF
 66:      enddo
 67:    enddo
 68:
 69:    status = TKclose( granuleHandleL2DPR )
 70:    STOP
 71:END
```

5.2.2 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
 1:F90= ifort
 2:MACHINE=LINUX
 3:FFLAGS=-g -CB -traceback -shared-intel -mcmodel=medium -fPIC -fpp -D$(MACHINE)
-DLANGUAGE_FORTRAN -DWRITEPIAINFO
 4:
 5:MAIN=./sample_L2_DPR_F
 6:
 7:OBJS= $(MAIN).o
 8:
 9:INC = -I$(HDF4_INC) ¥
10:     -I$(HDF5_INC) ¥
11:     -I$(TKIO)/inc/fortcode¥
12:     -I$(SZIP_INC)
13:
14:LIB = -L$(HDF4_LIB) ¥
15:     -L$(HDF5_LIB) ¥
16:     -L$(TKIO)/lib ¥
17:     -L$(SZIP_LIB)
18:
19:LIBES = -ltkcselect -ltkchdf4algs -ltkchdf4 ¥
20:     -ltkchdf5algs -ltkchdf5 -ltkctkTSDIS -ltkchelper¥
21:     -ltkc -lm -lmfhdf -ldf -lhdf5_hl -lhdf5 -ljpeg -lsz -lz -lxml2
22:
23:$(MAIN): $(OBJS)
24:     $(F90) $(FFLAGS) -o $(MAIN) $(OBJS) $(INC) $(LIB) $(LIBES)
25:
26:$(MAIN).o: $(MAIN).f90
27:     $(F90) $(FFLAGS) -c $(MAIN).f90 $(INC) $(LIB) $(LIBES)
28:clean:
29:     rm -f *.o $(MAIN)
```

5.2.1 のプログラムの名前です。

HDF4/HDF5 のインクルードディレクトリ、ライブラリディレクトリのパスです。PPS Toolkit(TKIO)を使用する場合、HDF4 も必要になるようです。

PPS Toolkit(TKIO)の Fortran のヘッダファイルがあるディレクトリのパスです

PPS Toolkit(TKIO)のライブラリがあるディレクトリのパスです

5.2.3 実行結果

5.2.1 で説明したプログラムの実行結果を示します。

第 1 引数の”bb”はジョブ名です。(任意の文字列で構いません)

第 2 引数の”/DPR/STD/…….HDF5”は HDF5 ファイル名です。

```
$ ./sample_L2_DPR_F "bb" "/DPR/STD/L2/2A.GPM.DPR.sample.HDF5"
 scan=        3946 ,angle=          19
 lat=   64.8442993164062      lon=  -163.994125366211
 precipEsurf=  0.0000000E+00 (mm/h)
$
```

5.3 L3 データ読み込み

5.3.1 ソースプログラム

以下は L3DPR 読み込みプログラム例です。ジョブ名と、L3DPR の HDF5 ファイル名を引数として、引数として指定されたファイルから、precipRateESurface.count というデータを読み込んでいます。

```
 1:PROGRAM SAMPLE
 2:
 3:#include "TKHEADERS.h"
 4:#include "TK_3DPR.h"
 5:#include "tkiofortdeclare.h"
 6:
 7:RECORD /TKINFO/ granuleHandle3DPR
 8:RECORD /L3DPR_GRIDS/ L3DPR
 9:
10:    INTEGER status
11:    INTEGER st, rt, chn, lnL, ltL
12:    CHARACTER*255 jobname, file
13:    REAL*4 precipEsurf(28,72,5,3,3)
14:
15:    call getarg( 1, jobname )
16:    call getarg( 2, file )
17:
18:    ! Open the file for reading.
19:    status = TKopen( file, '3DPR', TKREAD, 'HDF5', jobname, granuleHandle3DPR, 0 )
20:    IF ( status .NE. TK_SUCCESS ) THEN
21:      status = TKmessage( granuleHandle3DPR.jobname, TKERROR,'message to print' )
22:    ENDIF
23:
24:    ! Grid data read
25:    status = TKreadGrid( granuleHandle3DPR, L3DPR )
26:
27:    do ltL=1, 28
28:      do lnL=1, 72
29:        do chn=1, 5
30:          do rt=1, 3
31:            do st=1, 3
32:              ! precipRateESurface.mean Read
33:              precipEsurf(ltL, lnL, chn, rt, st) =
L3DPR.G1.precipRateESurface.mean(ltL, lnL, chn, rt, st)
34:
35:              ! print the value
36:              IF(lnL .eq. 72 .and. ltL .eq. 28 .and. chn .eq. 5 ) THEN
37:                write(*,*) "st=",st-1,"rt=",rt-1
38:                write(*,*) "chn=4 lnL=71 ltL=27
precipRateEsurface.mean=",precipEsurf(ltL, lnL, chn, rt, st), "(mm/h)"
39:              ENDIF
40:            enddo
41:          enddo
```

該当するアルゴリズム ID のヘッダファイルをインクルードします(表 5.1 プロダクトとアルゴリズム ID、TKIO ヘッダファイルの対応を参照)。L3DPR はアルゴリズム ID が"3DPR"なので"TK_3DPR.h"をインクルードします。

Fortran の場合必ずインクルードします。

granuleHandle3DPR は HDF5 ファイルの情報を格納する構造体で PPS Toolkit を使用する際に使用します。

L3DPR は HDF5 ファイルの構造体です。グリッドデータを格納します。

HDF5 ファイルのオープン
file:HDF5 ファイル名(引数で指定した文字列)
3DPR:アルゴリズム ID、 TKREAD:読み込み指定
HDF5:フォーマットタイプ
jobname:ジョブ名(引数で指定した文字列)
granuleHandle3DPR：ファイルポインタ(TKINFO 構造体を指定)
1 ：ファイルの内部圧縮(1 を指定)

グリッドデータ読み込み
granuleHandle3DPR：ファイルポインタ(TKINFO 構造体を指定)
L3DPR: 読み出したデータを格納する領域

precipRateESurface.mean 読み込み

正しく読み込めているか確認するため、一部分（このケースは、経度グリッド間隔が 72 番目、緯度グリッド間隔が 28 番目で、チャネルが 5 番目のデータ）を出力しています。

```
42:        enddo
43:      enddo
44:    enddo
45:
46:    ! HDF file close
47:    status = TKclose( granuleHandle3DPR )
48:    STOP
49:END
```

HDF ファイルのクローズ
granuleHandle3DPR：ファイルポインタ(TKINFO 構造体を指定)

5.3.2 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
 1:F90= ifort
 2:MACHINE=LINUX
 3:FFLAGS=-g -CB -traceback -shared-intel -mcmodel=medium -fPIC -fpp -D$(MACHINE)
-DLANGUAGE_FORTRAN -DWRITEPIAINFO
 4:
 5:MAIN=./sample_L3_DPR_F
 6:
 7:OBJS= $(MAIN).o
 8:
 9:INC = -I$(HDF4_INC) ¥
10:    -I$(HDF5_INC) ¥
11:    -I$(TKIO)/inc/fortcode¥
12:    -I$(SZIP_INC)
13:
14:LIB = -L$(HDF4_LIB) ¥
15:    -L$(HDF5_LIB) ¥
16:    -L$(TKIO)/lib ¥
17:    -L$(SZIP_LIB)
18:
19:LIBES = -ltkcselect -ltkchdf4algs -ltkchdf4 ¥
20:    -ltkchdf5algs -ltkchdf5 -ltkctkTSDIS -ltkchelper¥
21:    -ltkc -lm -lmfhdf -ldf -lhdf5_hl -lhdf5 -ljpeg -lsz -lz -lxml2
22:
23:$(MAIN): $(OBJS)
24:    $(F90) $(FFLAGS) -o $(MAIN) $(OBJS) $(INC) $(LIB) $(LIBES)
25:
26:$(MAIN).o: $(MAIN).f90
27:    $(F90) $(FFLAGS) -c $(MAIN).f90 $(INC) $(LIB) $(LIBES)
28:clean:
29:    rm -f *.o $(MAIN)
```

5.5.1 のプログラムの名前です。

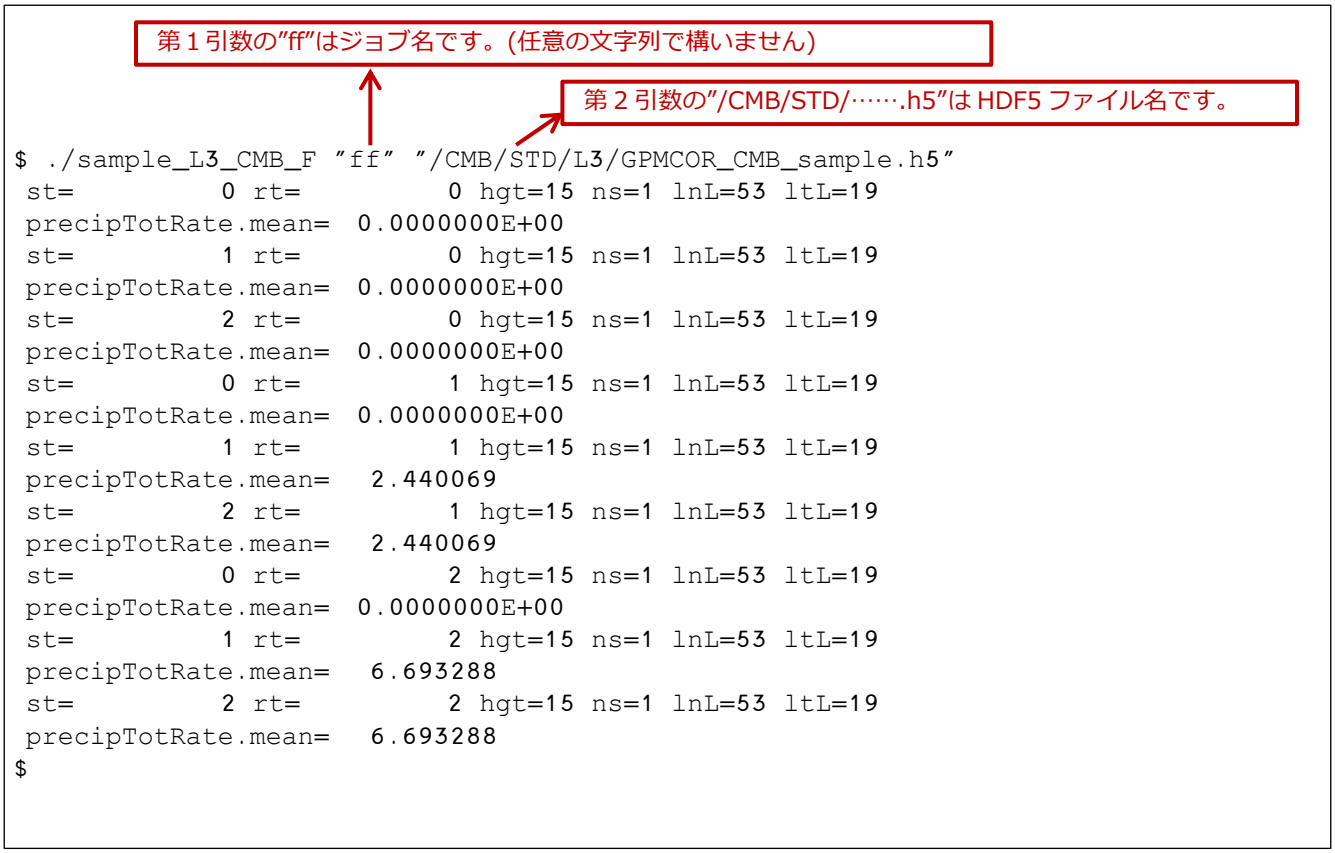
HDF4/HDF5 のインクルードディレクトリ、ライブラリディレクトリのパスです。PPS Toolkit(TKIO)を使用する場合、HDF4 も必要になるようです。

PPS Toolkit(TKIO)の Fortran 用のヘッダファイルがあるディレクトリのパスです

PPS Toolkit(TKIO)のライブラリがあるディレクトリのパスです

5.3.3 実行結果

5.3.1 で説明したプログラムの実行結果を示します。




```
$ ./sample_L3_CMB_F "ff" "/CMB/STD/L3/GPMCOR_CMB_sample.h5"
 st=          0 rt=          0 hgt=15 ns=1 lnL=53 ltL=19
 precipTotRate.mean=  0.0000000E+00
 st=          1 rt=          0 hgt=15 ns=1 lnL=53 ltL=19
 precipTotRate.mean=  0.0000000E+00
 st=          2 rt=          0 hgt=15 ns=1 lnL=53 ltL=19
 precipTotRate.mean=  0.0000000E+00
 st=          0 rt=          1 hgt=15 ns=1 lnL=53 ltL=19
 precipTotRate.mean=  0.0000000E+00
 st=          1 rt=          1 hgt=15 ns=1 lnL=53 ltL=19
 precipTotRate.mean=   2.440069
 st=          2 rt=          1 hgt=15 ns=1 lnL=53 ltL=19
 precipTotRate.mean=   2.440069
 st=          0 rt=          2 hgt=15 ns=1 lnL=53 ltL=19
 precipTotRate.mean=  0.0000000E+00
 st=          1 rt=          2 hgt=15 ns=1 lnL=53 ltL=19
 precipTotRate.mean=   6.693288
 st=          2 rt=          2 hgt=15 ns=1 lnL=53 ltL=19
 precipTotRate.mean=   6.693288
$
```

5.4 GSMaP_HDF5 データ読み込み

5.4.1 ソースプログラム

以下のサンプルプログラムは、ジョブ名と、GSMaP の HDF5 ファイル名を引数として、引数として指定されたファイルから、hourlyPrecipRateGC というデータを読み込んでいます。

```
 1:PROGRAM SAMPLE
 2:
 3:#include "TKHEADERS.h"
 4:#include "TK_3GSMAPH.h"
 5:#include "tkiofortdeclare.h"

 6:
 7:RECORD /TKINFO/ granuleHandle3GSMAPH
 8:RECORD /L3GSMAPH_GRID/ L3GSMAPH
 9:
10:    INTEGER status
11:    INTEGER nlat, nlon
12:    CHARACTER*255 jobname, file
13:    REAL*4 hourlyPrecipRateGC(1800,3600)
14:
15:    call getarg( 1, jobname )
16:    call getarg( 2, file )
17:


18:    ! Open the file for reading.
19:    status = TKopen( file, '3GSMAPH', TKREAD, 'HDF5', jobname,
granuleHandle3GSMAPH, 0 )
20:    IF ( status .NE. TK_SUCCESS ) THEN
21:           status = TKmessage( granuleHandle3GSMAPH.jobname, TKERROR,'message to
print' )
22:    ENDIF
23:
24:    ! Grid data read
25:    status = TKreadGrid( granuleHandle3GSMAPH, L3GSMAPH )
26:
27:    do nlat=1, 1800
28:      do nlon=1, 3600
29:        ! hourlyPrecipRateGC Read
30:        hourlyPrecipRateGC(nlat, nlon) = L3GSMAPH.hourlyPrecipRateGC(nlat,nlon)
31:


32:        ! print the value
33:        IF(nlat .eq. 1800 .and. nlon .eq. 3600 ) THEN
34:          write(*,*) "nlat=",nlat-1,"nlon=",nlon-1
35:          write(*,*) "hourlyPrecipRateGC=",hourlyPrecipRateGC(nlat, nlon),
"(mm/h)"
36:        ENDIF
37:      enddo
38:    enddo
39:
40:    ! HDF file close
41:    status = TKclose( granuleHandle3GSMAPH )
42:    STOP
```

該当するアルゴリズム ID のヘッダファイルをインクルードします(表 5.1 プロダクトとアルゴリズム ID、TKIO ヘッダファイルの対応を参照)。GSMaP はアルゴリズム ID が" 3GSMAPH" なので"TK_ 3GSMAPH.h"をインクルードします。

granuleHandle3GSMAPH は HDF5 ファイルの情報を格納する構造体で PPS Toolkit を使用する際に使用します。

L3GSMAPH は HDF5 ファイルの構造体です。グリッドデータを格納します。

HDF5 ファイルのオープン
file:HDF5 ファイル名(引数で指定した文字列)
3GSMAPH:アルゴリズム ID
TKREAD:読み込み指定
HDF5:フォーマットタイプ
jobname:ジョブ名(引数で指定した文字列)
granuleHandle3GSMAPH：ファイルポインタ
(TKINFO 構造体を指定)
1 ：ファイルの内部圧縮(1 を指定)

グリッドデータ読み込み
granuleHandle3GSMAPH：ファイルポインタ(TKINFO 構造体を指定)
L3GSMAPH: 読み出したデータを格納する領域

hourlyPrecipRateGC 読み込み

正しく読み込めているか確認するため、一部分を出力しています。

HDF ファイルのクローズ
granuleHandle3GSMAPH：ファイルポインタ(TKINFO 構造体を指定)

5.4.2 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
 1:F90= ifort
 2:MACHINE=LINUX
 3:FFLAGS=-g -CB -traceback -shared-intel -mcmodel=medium -fPIC -fpp -D$(MACHINE)
-DLANGUAGE_FORTRAN -DWRITEPIAINFO
 4:
 5:MAIN=./sample_GSMaP_HDF5_F
 6:
 7:OBJS= $(MAIN).o
 8:
 9:INC = -I$(HDF4_INC) ¥
10:     -I$(HDF5_INC) ¥
11:     -I$(TKIO)/inc/fortcode¥
12:     -I$(SZIP_INC)
13:
14:LIB = -L$(HDF4_LIB) ¥
15:     -L$(HDF5_LIB) ¥
16:     -L$(TKIO)/lib ¥
17:     -L$(SZIP_LIB)
18:
19:LIBES = -ltkcselect -ltkchdf4algs -ltkchdf4 ¥
20:     -ltkchdf5algs -ltkchdf5 -ltkctkTSDIS -ltkchelper¥
21:     -ltkc -lm -lmfhdf -ldf -lhdf5_hl -lhdf5 -ljpeg -lsz -lz -lxml2
22:
23:$(MAIN): $(OBJS)
24:     $(F90) $(FFLAGS) -o $(MAIN) $(OBJS) $(INC) $(LIB) $(LIBES)
25:
26:$(MAIN).o: $(MAIN).f90
27:     $(F90) $(FFLAGS) -c $(MAIN).f90 $(INC) $(LIB) $(LIBES)
28:clean:
29:     rm -f *.o $(MAIN)
```

5.8.1 のプログラムの名前です。

HDF5/HDF5 のインクルードディレクトリ、ライブラリディレクトリのパスです。PPS Toolkit(TKIO)を使用する場合、HDF4 も必要になるようです。

PPS Toolkit(TKIO)の Fortran 用のヘッダファイルがあるディレクトリのパスです

PPS Toolkit(TKIO)のライブラリがあるディレクトリのパスです

5.4.3 実行結果

5.4.1 で説明したプログラムの実行結果を示します。

第 1 引数の”hh”はジョブ名です。(任意の文字列で構いません)

第 2 引数の”/GSMaP/MCD/…….h5”は HDF5 ファイル名です。

```
$ ./sample_GSMaP_HDF5_F ”hh” ”/GSMaP/MCD/STD/GPMMRG_MAP_sample.h5”
 nlat=       1799 nlon=       3599
 hourlyPrecipRateGC=  -9999.900     (mm/h)
$
```

5.5 PPS Toolkit(TKIO)のバージョンについて

インストールした PPS Toolkit(TKIO)のバージョンと、HDF5 ファイルを作成したバージョンが異なると正常に読み込めない場合があります。その場合、HDF5 ファイルのバージョンを調べ、バージョンに合わせたヘッダファイル、アルゴリズム ID にプログラムを変更する必要があります。

HDF5 ファイルのバージョンを調べるには PPS Viewer THOR を使用して HDF5 ファイルの FileInfo を読出し、「DataFormatVersion」と「TKCodeBuildVersion」の値を確認します。

DataFormatVersion=bk

TKCodeBuildVersion=2

の場合、バージョンは bk2 となります。

プログラムの変更は以下の 3 箇所です。

1) インクルードするヘッダファイル名
2) アルゴリズム ID
3) ヘッダファイルの参照箇所

ヘッダファイルとアルゴリズム ID の変更はバージョンの表記を追加します。ヘッダファイルが「TK_2ADPR.h」、アルゴリズム ID が「2ADPR」の場合、ヘッダファイルは「TK_2DPR_bk2.h」、アルゴリズム ID は「2ADPR_bk2」となります。

ヘッダファイルの変更に伴い、ヘッダファイルの内容を参照している箇所も変更後のヘッダファイルに合わせた内容に変更する必要があります。

以下に L2DPR データ読み込みサンプルプログラムの修正例を示します。

```
 1:PROGRAM SAMPLE
 2:
 3:#include "TKHEADERS.h"
 4:#include "TK_2ADPR.h"

 5:#include "tkiofortdeclare.h"
 6:

 7:RECORD /TKINFO/ granuleHandleL2DPR


 8:RECORD /L2ADPR_SWATHS/ L2ADPR
 9:
10:    PARAMETER( NANGLE=49, NRANGE=176 )
11:
12:    INTEGER status, numOfScan
13:    INTEGER iscan, iangle, irange
14:    CHARACTER*255 jobname, file
15:    REAL*8 lat(NANGLE), lon(NANGLE)
16:    REAL*4 precipEsurf(NANGLE)
17:    REAL*4 zFactorCor(NRANGE,NANGLE)
18:    INTEGER*2 year, msec, dayOfYear
19:    BYTE month, dayOfMon, hour, min, sec
20:    REAL*8 secOfDay
21:
```

インクルードするヘッダファイルで、バージョンに合わせて変更するのはこのファイルです。（TK_xxxx.h）
“TK_2ADPR_bk2.h”に変更します。

TK_2ADPR.h の内容を参照しているのはこの部分です。
“TK_2ADPR_bk2.h”の内容を調べて対応する定義の
“L2ADPR_SWATHS_bk2”に変更します。

```
 22:    call getarg( 1, jobname )
 23:    call getarg( 2, file )
 24:
 25:    !Open the file for reading.
 26:    status = TKopen( file, '2ADPR', TKREAD, 'HDF5', jobname, granuleHandleL2DPR,
0 )
 27:    IF ( status .NE. TK_SUCCESS ) THEN
 28:      status = TKmessage( granuleHandleL2DPR.jobname, TKERROR,'message to print' )
 29:    ENDIF
 30:
 31:    status = TKgetMetaInt( granuleHandleL2DPR, 'NS_SwathHeader',
'NumberScansGranule', numOfScan )
 32:
 33:    do iscan=1,numOfScan
 34:      status = TKreadScan( granuleHandleL2DPR, L2ADPR )
 35:
 36:      ! ScanTime Read
 37:      year      = L2ADPR.NS.ScanTime.Year
 38:      month     = L2ADPR.NS.ScanTime.Month
 39:      dayOfMon  = L2ADPR.NS.ScanTime.DayOfMonth
 40:      hour      = L2ADPR.NS.ScanTime.Hour
 41:      min       = L2ADPR.NS.ScanTime.Minute
 42:      sec       = L2ADPR.NS.ScanTime.Second
 43:      msec      = L2ADPR.NS.ScanTime.MilliSecond
 44:      dayOfYear = L2ADPR.NS.ScanTime.DayOfYear
 45:      secOfDay  = L2ADPR.NS.ScanTime.SecondOfDay
 46:
 47:      do iangle=1,NANGLE
 48:
 49:        ! Latitude and Longitude Read
 50:        lat(iangle)  = L2ADPR.NS.Latitude(iangle)
 51:        lon(iangle)  = L2ADPR.NS.Longitude(iangle)
 52:
 53:        ! precipRateESurface Read
 54:        precipEsurf(iangle) = L2ADPR.NS.SLV.precipRateESurface(iangle)
 55:

 56:        do irange=1,NRANGE
 57:          zFactorCor(irange, iangle) = L2ADPR.NS.SLV.zFactorCorrected(irange,
iangle)
 58:        enddo
 59:
 60:        ! print the value
 61:        IF(iscan .eq. 3947 .and. iangle .eq. 20 ) THEN
 62:          write(*,*) "scan=",iscan-1,",angle=",iangle-1
 63:          write(*,*) "lat=", lat(iangle), "lon=", lon(iangle)
 64:          write(*,*) "precipEsurf=",precipEsurf(iangle), "(mm/h)"
 65:        ENDIF
 66:      enddo
 67:    enddo
 68:
 69:    status = TKclose( granuleHandleL2DPR )
 70:    STOP
 71:END
```

アルゴリズム ID を指定しているのは、
この部分です。バージョンを追加して
“2ADPR_bk2”に変更します。

6. HDF ライブラリで GPM データ読み込み

HDF ライブラリを使用した Fortran プログラムの作成方法について説明します。

赤字の解説はサンプルプログラムについて説明しています。

青字の解説は HDF ライブラリまたは衛星基礎知識について説明しています。

6.1 L2DPR データ読み込み

6.1.1 ソースプログラム

以下のサンプルプログラムは、filename で指定されたファイルから、precipRateESurface というデータを読み込んでいます。

```
 1:PROGRAM SAMPLE
 2:
 3:       use hdf5   ! This module contains all necessary modules
 4:
 5:       CHARACTER(LEN=34), PARAMETER :: filename =
"/DPR/STD/L2/2A.GPM.DPR.sample.HDF5"
 6:
 7:       CHARACTER(LEN=26), PARAMETER :: dsetname = "/NS/SLV/precipRateESurface"
 8:
 9:       INTEGER(HID_T) :: file_id      ! File identifier
10:       INTEGER(HID_T) :: dset_id      ! Dataset identifier
11:       REAL*4, DIMENSION(49,7932) :: precipRateESurface ! Data buffers
12:       INTEGER(HSIZE_T), DIMENSION(2) :: data_dims
13:       INTEGER error
14:
15:       ! Initialize FORTRAN interface.
16:       CALL h5open_f(error)
17:
18:       ! Open an existing file.
19:       CALL h5fopen_f (filename, H5F_ACC_RDWR_F, file_id, error)
20:
```

読み込む HDF5 のファイル名です。

HDF5 ファイルの中から読み込むデータ名です。

HDF5 ライブラリ初期化
error：エラー情報(0:正常)

HDF5 ファイルオープン
filename：HDF5 ファイル名
H5F_ACC_RDWR_F：アクセス指定
file_id：ファイル ID(出力情報)
error：エラー情報(0:正常)

データセットのオープン
file_id：h5fopen_f で出力した値を指定します。
dsetname：読み込むデータの名前です。
dset_id：データセット ID(出力情報)
error：エラー情報(0:正常)

```
 21:       ! Open an existing dataset.
 22:       CALL h5dopen_f(file_id, dsetname, dset_id, error)
 23:
 24:       ! Read precipRateESurface.
```

データ読み込み
dset_id：h5dopen_f で出力した値を指定します。

```
 25:       data_dims(1) = 49
 26:       data_dims(2) = 7932
 27:       CALL h5dread_f(dset_id, H5T_NATIVE_REAL, precipRateESurface, data_dims,
error)
 28:
 29:       ! print the value
 30:       write(*,*) "filename=",filename,""
 31:       write(*,*) "dataset name=",dsetname,""
 32:       write(*,*) "precipRateESurface(9,3763)=",precipRateESurface(10,3764),
"(mm/h)"
 33:       write(*,*) "precipRateESurface(10,5635)=",precipRateESurface(11,5636),
"(mm/h)"
 34:
 35:       ! Close the dataset.
 36:       CALL h5dclose_f(dset_id, error)
 37:
 38:       ! Close the file.
 39:       CALL h5fclose_f(file_id, error)
 40:
 41:       ! Close FORTRAN interface.
 42:       CALL h5close_f(error)
 43:
 44:   STOP
 45:END
```

6.1.2 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
  1:F90= ifort
  2:MACHINE=LINUX
  3:FFLAGS=-g -CB -traceback -shared-intel -mcmodel=medium -fPIC -fpp -D$(MACHINE)
-DLANGUAGE_FORTRAN -DWRITEPIAINFO
  4:
  5:MAIN=./sample_HDF5_L2_DPR_F
  6:
  7:OBJS= $(MAIN).o
  8:HDF5_INC= /export/trmm5/tool/x86_64/hdf5-1.8.9_ifort/include
  9:HDF5_LIB= /export/trmm5/tool/x86_64/hdf5-1.8.9_ifort/lib
 10:
 11:INC = -I$(HDF5_INC) ¥
 12:      -I$(SZIP_INC)
 13:LIB = -L$(HDF5_LIB) ¥
 14:      -L$(SZIP_LIB)
 15:
 16:LIBES =   -lm -lhdf5_fortran -lhdf5  -lz
 17:
 18:$(MAIN): $(OBJS)
 19:   $(F90) $(FFLAGS) -o $(MAIN) $(OBJS) $(INC) $(LIB) $(LIBES)
 20:
 21:$(MAIN).o: $(MAIN).f90
 22:   $(F90) $(FFLAGS) -c $(MAIN).f90 $(INC) $(LIB) $(LIBES)
 23:clean:
 24:   rm -f *.o $(MAIN)
```

6.1.1 のプログラムの名前です。

HDF5/HDF5 のインクルードディレクトリ、ライブラリディレクトリのパスです。

6.1.3 実行結果

6.1.1 で説明したプログラムの実行結果を示します。

```
$ ./sample_HDF5_L2_DPR_F
 filename= /DPR/STD/L2/2A.GPM.DPR.sample.HDF5
 dataset name=/NS/SLV/precipRateESurface
 precipRateESurface(9,3763)=   2.790208     (mm/h)
 precipRateESurface(10,5635)=   2.090051     (mm/h)
$
```

6.2 L3DPR データ読み込み

6.2.1 ソースプログラム

以下のサンプルプログラムは、filename で指定されたファイルから、precipRateESurface というデータを読み込んでいます。

```
  1:PROGRAM SAMPLE
  2:
  3:    use hdf5   ! This module contains all necessary modules
  4:
  5:    CHARACTER(LEN=38), PARAMETER :: filename =
"/DPR/STD/L3/3A-MO.GPM.DPR.sample.HDF5"
  6:
  7:    CHARACTER(LEN=34), PARAMETER :: dsetname =
"/Grids/G1/precipRateESurface/mean"
  8:
  9:    INTEGER(HID_T) :: file_id      ! File identifier
 10:    INTEGER(HID_T) :: dset_id      ! Dataset identifier
 11:    REAL*4, DIMENSION(28,72,5,3,3) :: precipRateESurface ! Data buffers
 12:    INTEGER(HSIZE_T), DIMENSION(5) :: data_dims
 13:   INTEGER error
 14:
 15:    ! Initialize FORTRAN interface.
 16:    CALL h5open_f(error)
 17:
 18:    ! Open an existing file.
 19:    CALL h5fopen_f (filename, H5F_ACC_RDWR_F, file_id, error)
 20:
 21:    ! Open an existing dataset.
 22:    CALL h5dopen_f(file_id, dsetname, dset_id, error)
```

読み込む HDF5 のファイル名です。

HDF5 ファイルの中から読み込むデータ名です。

HDF5 ライブラリ初期化
error：エラー情報(0:正常)

HDF5 ファイルオープン
filename：HDF5 ファイル名
H5F_ACC_RDWR_F：アクセス指定
file_id：ファイル ID(出力情報)
error：エラー情報(0:正常)

データセットのオープン
file_id：h5fopen_f で出力した値を指定します。
dsetname：読み込むデータの名前です。
dset_id：データセット ID(出力情報)
error：エラー情報(0:正常)

```
 23:
 24:    ! Read precipRateESurface.
 25:    data_dims(1) = 28
 26:    data_dims(2) = 72
 27:    data_dims(3) = 5
 28:    data_dims(4) = 3
 29:    data_dims(5) = 3
 30:    CALL h5dread_f(dset_id, H5T_NATIVE_REAL, precipRateESurface, data_dims,
error)
 31:
 32:    ! print the value
 33:    write(*,*) "filename=",filename
 34:    write(*,*) "dataset name=",dsetname
 35:    write(*,*)
"precipRateESurface.mean(0,0,0,0,0)=",precipRateESurface(1,1,1,1,1)
 36:    write(*,*)
"precipRateESurface.mean(27,71,4,2,2)=",precipRateESurface(28,72,5,3,3)
 37:
 38:    ! Close the dataset.
 39:    CALL h5dclose_f(dset_id, error)
 40:
 41:    ! Close the file.
 42:    CALL h5fclose_f(file_id, error)
 43:
 44:    ! Close FORTRAN interface.
 45:    CALL h5close_f(error)
 46:
 47:   STOP
 48:END
```

データ読み込み
dset_id：h5dopen_f で出力した値を指定します。

6.2.2 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
  1:F90= ifort
  2:MACHINE=LINUX
  3:FFLAGS=-g -CB -traceback -shared-intel -mcmodel=medium -fPIC -fpp -D$(MACHINE)
-DLANGUAGE_FORTRAN -DWRITEPIAINFO
  4:
  5:MAIN=./sample_HDF5_L3_DPR_F
  6:
  7:OBJS= $(MAIN).o
  8:HDF5_INC= /export/trmm5/tool/x86_64/hdf5-1.8.9_ifort/include
  9:HDF5_LIB= /export/trmm5/tool/x86_64/hdf5-1.8.9_ifort/lib
 10:
 11:INC = -I$(HDF5_INC) ¥
 12:      -I$(SZIP_INC)
 13:LIB = -L$(HDF5_LIB) ¥
 14:      -L$(SZIP_LIB)
 15:
 16:LIBES =   -lm -lhdf5_fortran -lhdf5  -lz
 17:
 18:$(MAIN): $(OBJS)
 19:   $(F90) $(FFLAGS) -o $(MAIN) $(OBJS) $(INC) $(LIB) $(LIBES)
 20:
 21:$(MAIN).o: $(MAIN).f90
 22:   $(F90) $(FFLAGS) -c $(MAIN).f90 $(INC) $(LIB) $(LIBES)
 23:clean:
 24:   rm -f *.o $(MAIN)
```

6.2.1 のプログラムの名前です。

HDF5/HDF5 のインクルードディレクトリ、ライブラリディレクトリのパスです。

6.2.3 実行結果

6.2.1 で説明したプログラムの実行結果を示します。

```
$ ./sample_HDF5_L3_DPR_F
 filename=
 /DPR/STD/L3/3A-MO.GPM.DPR.sample.HDF5
 dataset name=/Grids/G1/precipRateESurface/mean
 precipRateESurface.mean(0,0,0,0,0)=  0.4883168
 precipRateESurface.mean(27,71,4,2,2)=   1.369588
$
```

6.3 GSMaP_HDF5 データ読み込み

6.3.1 ソースプログラム

以下のサンプルプログラムは、filename で指定されたファイルから、precipRateESurface というデータを読み込んでいます。

```
   1:PROGRAM SAMPLE
   2:
   3:    use hdf5   ! This module contains all necessary modules
   4:
   5:    CHARACTER(LEN=35), PARAMETER :: filename =
"/GSMaP/MCD/STD/GPMMRG_MAP_sample.h5"
   6:
   7:    CHARACTER(LEN=24), PARAMETER :: dsetname = "/Grid/hourlyPrecipRateGC"
   8:
   9:    INTEGER(HID_T) :: file_id      ! File identifier
  10:    INTEGER(HID_T) :: dset_id      ! Dataset identifier
  11:    REAL*4, DIMENSION(1800,3600) :: hourlyPrecipRateGC ! Data buffers
  12:    INTEGER(HSIZE_T), DIMENSION(2) :: data_dims
  13:   INTEGER error
  14:
  15:    ! Initialize FORTRAN interface.
  16:    CALL h5open_f(error)
  17:
  18:    ! Open an existing file.
  19:    CALL h5fopen_f (filename, H5F_ACC_RDWR_F, file_id, error)
  20:
  21:    ! Open an existing dataset.
  22:    CALL h5dopen_f(file_id, dsetname, dset_id, error)
  23:
  24:    ! Read hourlyPrecipRateGC.
  25:    data_dims(1) = 1800
  26:    data_dims(2) = 3600
  27:    CALL h5dread_f(dset_id, H5T_NATIVE_REAL, hourlyPrecipRateGC, data_dims,
error)
  28:
```

読み込む HDF5 のファイル名です。

HDF5 ファイルの中から読み込むデータ名です。

HDF5 ライブラリ初期化
error : エラー情報(0:正常)

HDF5 ファイルオープン
filename : HDF5 ファイル名
H5F_ACC_RDWR_F : アクセス指定
file_id : ファイル ID(出力情報)
error : エラー情報(0:正常)

データセットのオープン
file_id : h5fopen_f で出力した値を指定します。
dsetname : 読み込むデータの名前です。
dset_id : データセット ID(出力情報)
error : エラー情報(0:正常)

データ読み込み
dset_id : h5dopen_f で出力した値を指定します。

```
 29:    ! print the value
 30:    write(*,*) "filename=",filename,""
 31:    write(*,*) "dataset name=",dsetname,""
 32:    write(*,*) "hourlyPrecipRateGC(0,0)=",hourlyPrecipRateGC(1,1), "(mm/h)"
 33:    write(*,*) "hourlyPrecipRateGC(1799,3599)=",hourlyPrecipRateGC(1800,3600),
"(mm/h)"
 34:
 35:    ! Close the dataset.
 36:    CALL h5dclose_f(dset_id, error)
 37:
 38:    ! Close the file.
 39:    CALL h5fclose_f(file_id, error)
 40:
 41:    ! Close FORTRAN interface.
 42:    CALL h5close_f(error)
 43:
 44:   STOP
 45:END
```

6.3.2 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
  1:F90= ifort
  2:MACHINE=LINUX
  3:FFLAGS=-g -CB -traceback -shared-intel -mcmodel=medium -fPIC -fpp -D$(MACHINE)
-DLANGUAGE_FORTRAN -DWRITEPIAINFO
  4:
  5:MAIN=./sample_HDF5_GSMaP_F
  6:
  7:OBJS= $(MAIN).o
  8:HDF5_INC= /export/trmm5/tool/x86_64/hdf5-1.8.9_ifort/include
  9:HDF5_LIB= /export/trmm5/tool/x86_64/hdf5-1.8.9_ifort/lib
 10:
 11:INC = -I$(HDF5_INC) ¥
 12:     -I$(SZIP_INC)
 13:LIB = -L$(HDF5_LIB) ¥
 14:     -L$(SZIP_LIB)
 15:
 16:LIBES =   -lm -lhdf5_fortran -lhdf5  -lz
 17:
 18:$(MAIN): $(OBJS)
 19:   $(F90) $(FFLAGS) -o $(MAIN) $(OBJS) $(INC) $(LIB) $(LIBES)
 20:
 21:$(MAIN).o: $(MAIN).f90
 22:   $(F90) $(FFLAGS) -c $(MAIN).f90 $(INC) $(LIB) $(LIBES)
 23:clean:
 24:   rm -f *.o $(MAIN)
```

6.3.1 のプログラムの名前です。

HDF5/HDF5 のインクルードディレクトリ、ライブラリディレクトリのパスです。

6.3.3 実行結果

6.3.1 で説明したプログラムの実行結果を示します。

```
$ ./sample_HDF5_GSMaP_F
 filename=/GSMaP/MCD/STD/GPMMRG_MAP_sample.h5
 dataset name=/Grid/hourlyPrecipRateGC
 hourlyPrecipRateGC(0,0)=  -9999.900     (mm/h)
 hourlyPrecipRateGC(1799,3599)=  -9999.900     (mm/h)
$
```

7. h5dump で GPM データ読み込み

h5dump を使用して、HDF5 ファイルから読み込みたいデータのバイナリファイルを作成し、そのバイナリファイルを読み出す Fortran プログラムの作成方法について説明します。

赤字の解説はサンプルプログラムについて説明しています。

青字の解説は HDF ライブラリまたは衛星基礎知識について説明しています。

7.1 L2 データ読み込み

7.1.1 バイナリファイルの作成

h5dump を使用してバイナリファイルの作成を行うシェルの作成例を示します。

```
  1:#!/bin/tcsh -f
  2:
  3:set h5dump_bin=/home/tool/hdf5-1.8.9/bin/h5dump
  4:set file_dir=/DPR/STD/L2/
  5:set OUTPUT=/h5dump_samplecode/L2/binfile/
  6:
  7:cd ${file_dir}
  8:set file_in=(`ls *008435* |sed 's/.HDF5/ /' `)
  9:mkdir -p ${OUTPUT}
 10:cd ${OUTPUT}
 11:
 12:foreach file ($file_in)
 13:echo ${file}
 14:$h5dump_bin -d NS/SLV/precipRateESurface -b -o ${file}.precipRateESurface
${file_dir}/${file}.HDF5
 15:end
```

h5dump のフルパスを指定しています。

HDF5 ファイルが存在するディレクトリを指定しています。

バイナリファイルの出力先のディレクトリを指定しています。

HDF5 ファイルが存在するディレクトリで処理するファイルを絞込む場合指定します。"*008435*"はファイル名に"008435"が含まれているファイルのみ対象とする意味です。

ファイル名の拡張子の部分を削除しています。

対象ファイル数分ループ

バイナリファイル作成
読み込んだファイルから-d で指定されたデータを、-b-o で指定されたファイル名でバイナリファイルを作成しています。

上記のシェルを実行すると以下のように表示され、OUTPUT で指定したディレクトリにバイナリファイルが作成されます。

```
$ ./dump_L2.sh
2A.GPM.DPR.sample.HDF5 "/DPR/STD/L2//2A.GPM.DPR.sample.HDF5" {
DATASET "NS/SLV/precipRateESurface" {
   DATATYPE  H5T_IEEE_F32LE
   DATASPACE  SIMPLE { ( 7932, 49 ) / ( H5S_UNLIMITED, 49 ) }
   DATA {
   }
}
}
```

7.1.2 ソースプログラム

以下のサンプルプログラムは、inputfile で指定されたバイナリファイルから情報を読み込んでいます。

```
   1:program sample
   2:
   3:    CHARACTER(LEN=66), PARAMETER :: filename
="/h5dump_samplecode/L2/binfile/2A.GPM.DPR.sample.precipRateESurface"
   4:
   5:    ! read data area
   6:    real*4 precipRateESurface(49,7932)
   7:
   8:    ! binary file open
   9:    open(10,file=filename,access='direct',status='old',recl=4*49*7932)
  10:
  11:    ! binary file read
  12:    read(10,rec=1) precipRateESurface
  13:
  14:    ! print the value
  15:    write(*,*) "filename=",filename,""
  16:    write(*,*) "precipRateESurface(9,3763)=",precipRateESurface(10,3764),
"(mm/h)"
  17:    write(*,*) "precipRateESurface(10,5635)=",precipRateESurface(11,5636),
"(mm/h)"
  18:
  19:    ! binary file close
  20:    close(10)
  21:
  22:end
```

7.1.1 で作成したバイナリファイルを指定しています。

バイナリファイルのオープン
recl：レコード長(precipRateESurface のサイズを指定)

バイナリファイルの読み込み
1 回で全データを precipRateESurface に読み込んでいます。

7.1.3 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
  1:F90= ifort
  2:MACHINE=LINUX
  3:FFLAGS=-g -CB -traceback -shared-intel -mcmodel=medium -fPIC -fpp -D$(MACHINE)
-DLANGUAGE_FORTRAN -DWRITEPIAINFO
  4:
  5:MAIN=./sample_h5dump_L2_F
  6:
  7:OBJS= $(MAIN).o
  8:
  9:INC = -I$(HDF5_INC) ¥
 10:   -I$(SZIP_INC)
 11:
 12:LIB = -L$(HDF5_LIB) ¥
 13:   -L$(SZIP_LIB)
 14:
 15:LIBES =   -lm -ljpeg -lz -lxml2
 16:
 17:$(MAIN): $(OBJS)
 18:   $(F90) $(FFLAGS) -o $(MAIN) $(OBJS) $(INC) $(LIB) $(LIBES)
 19:
 20:$(MAIN).o: $(MAIN).f90
 21:   $(F90) $(FFLAGS) -c $(MAIN).f90 $(INC) $(LIB) $(LIBES)
 22:clean:
 23:   rm -f *.o $(MAIN)
```

5:MAIN=./sample_h5dump_L2_F → 7.1.1 のプログラムの名前です。

9:INC = -I$(HDF5_INC) ¥ / 10: -I$(SZIP_INC) → HDF5/HDF5 のインクルードディレクトリ、ライブラリディレクトリのパスです。

7.1.4 実行結果

7.1.2 で説明したプログラムの実行結果を示します。

```
$ ./sample_h5dump_L2_F
 filename=/h5dump_samplecode/L2/binfile/2A.GPM.DPR.sample.precipRateESurface
 precipRateESurface(9,3763)=   2.790208     (mm/h)
 precipRateESurface(10,5635)=   2.090051     (mm/h)
$
```

7.2 L3 データ読み込み

7.2.1 バイナリファイルの作成

h5dump を使用してバイナリファイルの作成を行うシェルの作成例を示します。

```
 1:#!/bin/tcsh -f
 2:
 3:set h5dump_bin=/home/tool/hdf5-1.8.9/bin/h5dump
 4:set file_dir=/DPR/STD/L3/
 5:set OUTPUT=/h5dump_samplecode/L3/binfile/
 6:
 7:cd ${file_dir}
 8:set file_in=(`ls *150801* |sed 's/.HDF5/ /' `)
 9:cd ${OUTPUT}
10:
11:foreach file ($file_in)
12:echo ${file}
13:$h5dump_bin -d /Grids/G1/precipRateESurface/mean -b -o
${file}.precipRateESurface_mean ${file_dir}/${file}.HDF5
14:end
```

h5dump のフルパスを指定しています。

HDF5 ファイルが存在するディレクトリを指定しています。

バイナリファイルの出力先のディレクトリを指定しています。

HDF5 ファイルが存在するディレクトリで処理するファイルを絞込む場合指定します。"*150801*"はファイル名に"150801"が含まれているファイルのみ対象とする意味です。

ファイル名の拡張子の部分を削除しています。

対象ファイル数分ループ

バイナリファイル作成
読み込んだファイルから-d で指定されたデータを、-b-o で指定されたファイル名でバイナリファイルを作成しています。

上記のシェルを実行すると以下のように表示され、OUTPUT で指定したディレクトリにバイナリファイルが作成されます。

```
$ ./dump_L3.sh
3A-MO.GPM.DPR.HDF5 "/DPR/STD/L3//3A-MO.GPM.DPR.sample.HDF5" {
DATASET "/Grids/G1/precipRateESurface/mean" {
   DATATYPE  H5T_IEEE_F32LE
   DATASPACE  SIMPLE { ( 3, 3, 5, 72, 28 ) / ( 3, 3, 5, 72, 28 ) }
   DATA {
   }
}
}
$
```

7.2.2 ソースプログラム

以下のサンプルプログラムは、inputfile で指定されたバイナリファイルから情報を読み込んでいます。

```
   1:program sample
   2:
   3:    CHARACTER(LEN=74), PARAMETER :: filename
="/h5dump_samplecode/L3/binfile/3A-MO.GPM.DPR.sample.precipRateESurface_mean"
   4:
   5:    ! read data area
   6:    real*4 precipRateESurface(28,72,5,3,3)
   7:
   8:    ! binary file open
   9:    open(10,file=filename,access='direct',status='old',recl=4*28*72*5*3*3)
  10:
  11:    ! binary file read
  12:    read(10,rec=1) precipRateESurface
  13:
  14:    ! print the value
  15:    write(*,*) "filename=",filename
  16:    write(*,*) "precipRateESurface.mean(0,0,0,0,0)=",
precipRateESurface(1,1,1,1,1)
  17:    write(*,*) "precipRateESurface.mean(27,71,4,2,2)=",
precipRateESurface(28,72,5,3,3)
  18:
  19:    ! binary file close
  20:    close(10)
  21:
  22:end
```

7.2.1 で作成したバイナリファイルを指定しています。

バイナリファイルのオープン
recl：レコード長(precipRateESurface のサイズを指定)

バイナリファイルの読み込み
1 回で全データを precipRateESurface に読み込んでいます。

7.2.3 コンパイル方法

コンパイル時に使用する makefile の例を説明します。

```
  1:F90= ifort
  2:MACHINE=LINUX
  3:FFLAGS=-g -CB -traceback -shared-intel -mcmodel=medium -fPIC -fpp -D$(MACHINE)
-DLANGUAGE_FORTRAN -DWRITEPIAINFO
  4:
  5:MAIN=./sample_h5dump_L3_F
  6:
  7:OBJS= $(MAIN).o
  8:
  9:INC = -I$(HDF5_INC) ¥
 10:   -I$(SZIP_INC)
 11:
 12:LIB = -L$(HDF5_LIB) ¥
 13:   -L$(SZIP_LIB)
 14:
 15:LIBES =  -lm -ljpeg -lz -lxml2
 16:
 17:$(MAIN): $(OBJS)
 18:   $(F90) $(FFLAGS) -o $(MAIN) $(OBJS) $(INC) $(LIB) $(LIBES)
 19:
 20:$(MAIN).o: $(MAIN).f90
 21:   $(F90) $(FFLAGS) -c $(MAIN).f90 $(INC) $(LIB) $(LIBES)
 22:clean:
 23:   rm -f *.o $(MAIN)
```

7.2.1 のプログラムの名前です。

HDF5/HDF5 のインクルードディレクトリ、ライブラリディレクトリのパスです。

7.2.4 実行結果

7.2.2 で説明したプログラムの実行結果を示します。

```
$ ./sample_h5dump_L3_F
 filename=
/h5dump_samplecode/L3/binfile/3A-MO.GPM.DPR.sample.precipRateESurface_mean
 precipRateESurface.mean(0,0,0,0,0)=  0.4883168
 precipRateESurface.mean(27,71,4,2,2)=   1.369588
$
```

改版履歴

版数	日付	改版内容	備考
1	2016/1/26		
2	2017/9/13	1. はじめに : 表 1.1 に python の記載を追加、それに伴いフローチャート修正。 表 1.2 サンプルコード動作確認表を追加。 4. ライブラリ・ツールのインストール:表 4.2 プロダクトバージョンと PPS Toolkit(TKIO)の対応バージョンを追加。 表 4.3 動作環境の tkio-3.70.7 と記載している箇所を tkio-x.xx.x に変更 4.2.4 環境設定ファイルの編集: tkio-3.70.7 と記載している箇所を tkio-x.xx.x に変更。	
3	2018/3/15	2. 関連文書、サンプルプログラムの入手方法 : 表 3.1 サンプルプログラム一覧を追加	