GPM/TRMM Data reading program guide (FORTRAN version)

2022/02/04

8th ed.

This document describes how to create a program (FORTRAN) to read data from the Global Precipitation Measurement (GPM) satellite.

The sample programs described in this document have been tested with product version 07 for GPM/TRMM and with product version 05 for GSMaP.

Table of Contents

Introduction

This document describes how to read in GPM/TRMM data using the FORTRAN language.
The GPM and TRMM formats have been unified since version 06 products (equivalent to TRMM version 8), and the latest algorithm is version 07 (equivalent to TRMM version 9). The latest algorithm is version 07 (equivalent to TRMM version 9), which can be read in the same way in this sample program.
In addition to FORTRAN, there are other methods to read GPM data as shown in Table 1.1. Please refer to the "Read Method Decision Flow" on the next page to determine which method to use.
Table 1.2 lists the operating systems on which the sample programs used in this document were tested.

Table 1.1 Data loading methods

	Data loading method	Name of material	remarks
1	Using THOR	GPM/TRMM Data Loading Program Guide (THOR Edition)	
2	Use IDL	GPM/TRMM Data Loading Program Guide (IDL version)	
3	Use C	GPM/TRMM data reading program guide (C language version)	
4	Using FORTRAN	GPM/TRMM Data Loading Program Guide (FORTRAN Edition)	
5	Using Python	GPM/TRMM data reading program guide (Python version)	

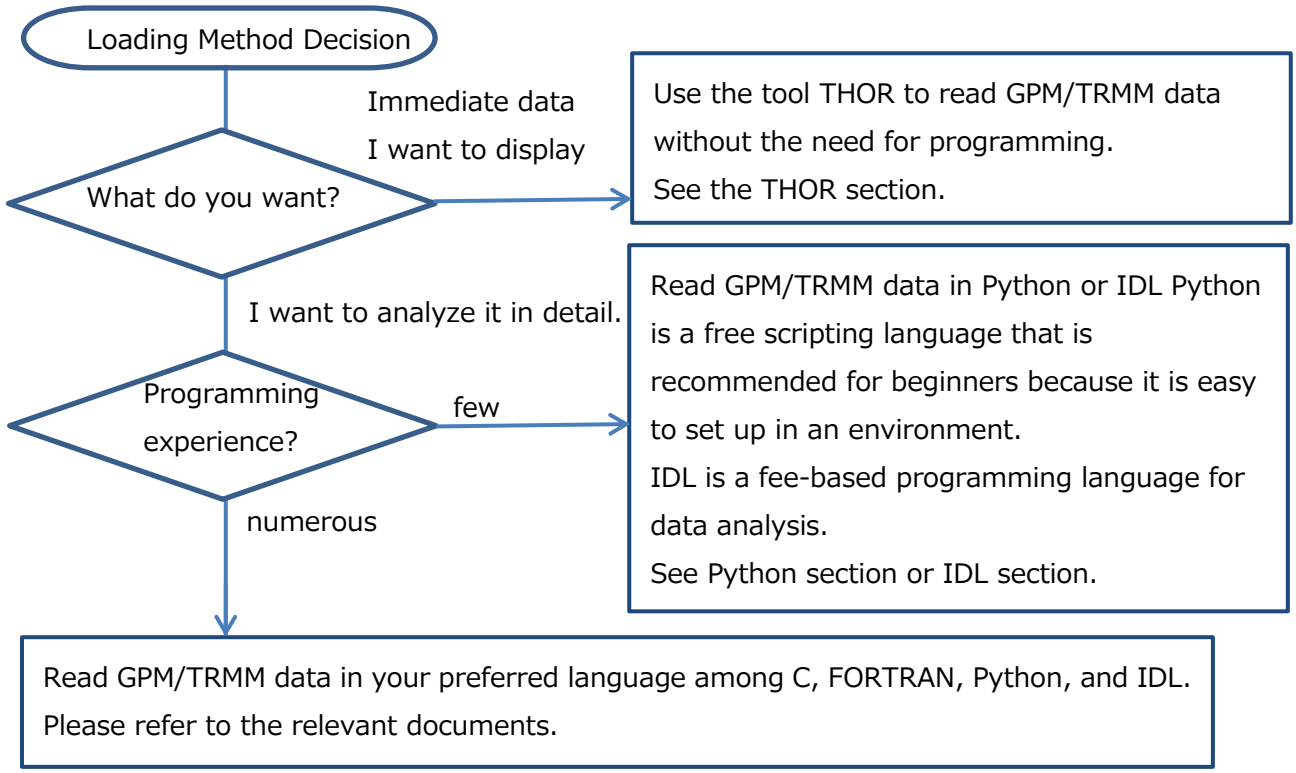


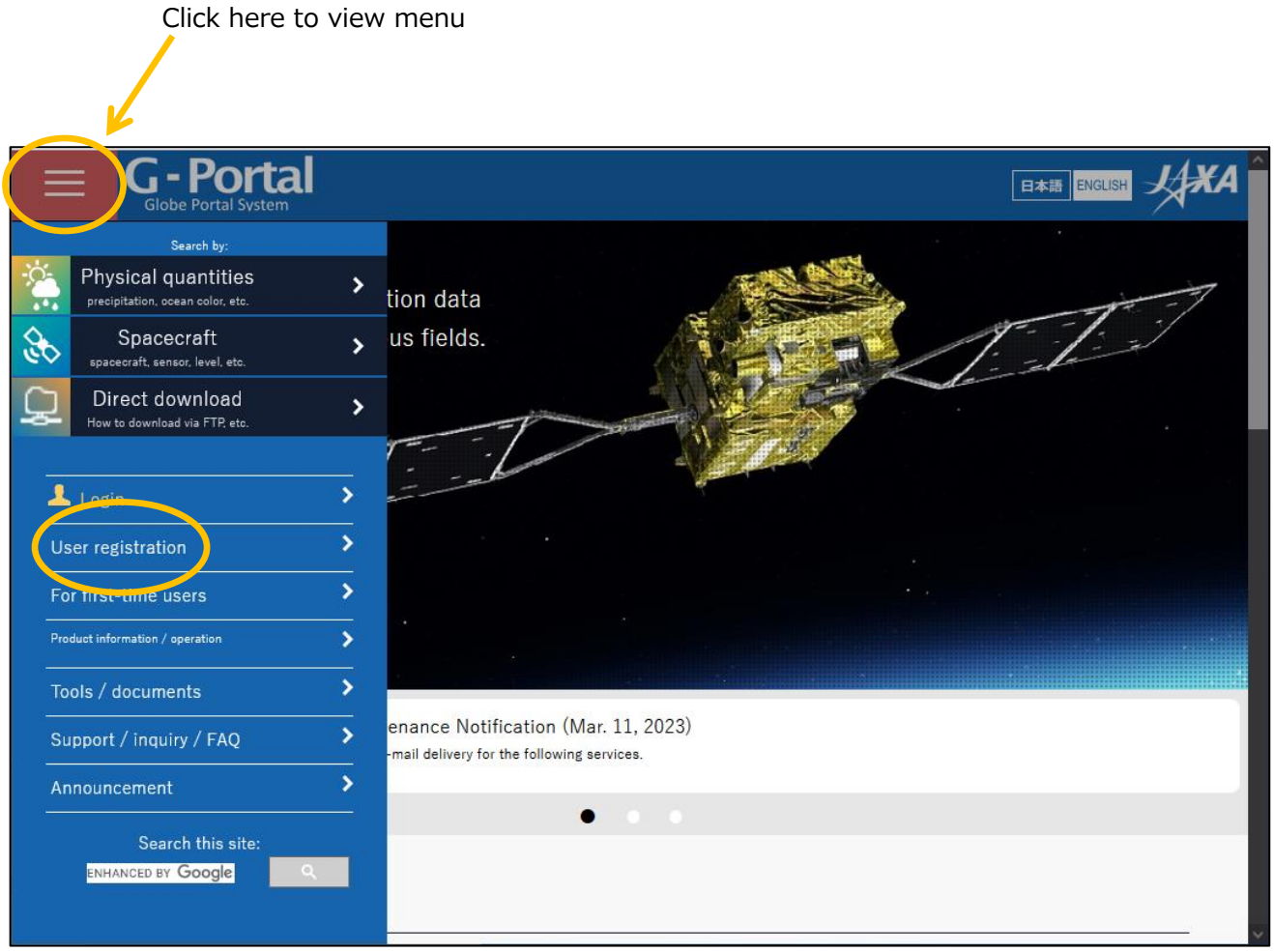
Table 1.2 Sample Program Operation Check Table

	sample program	Linux	Windows	remarks
1	c	○	-	
2	FORTRAN	○	-	
3	Python	○	○	
4	IDL	○	○	

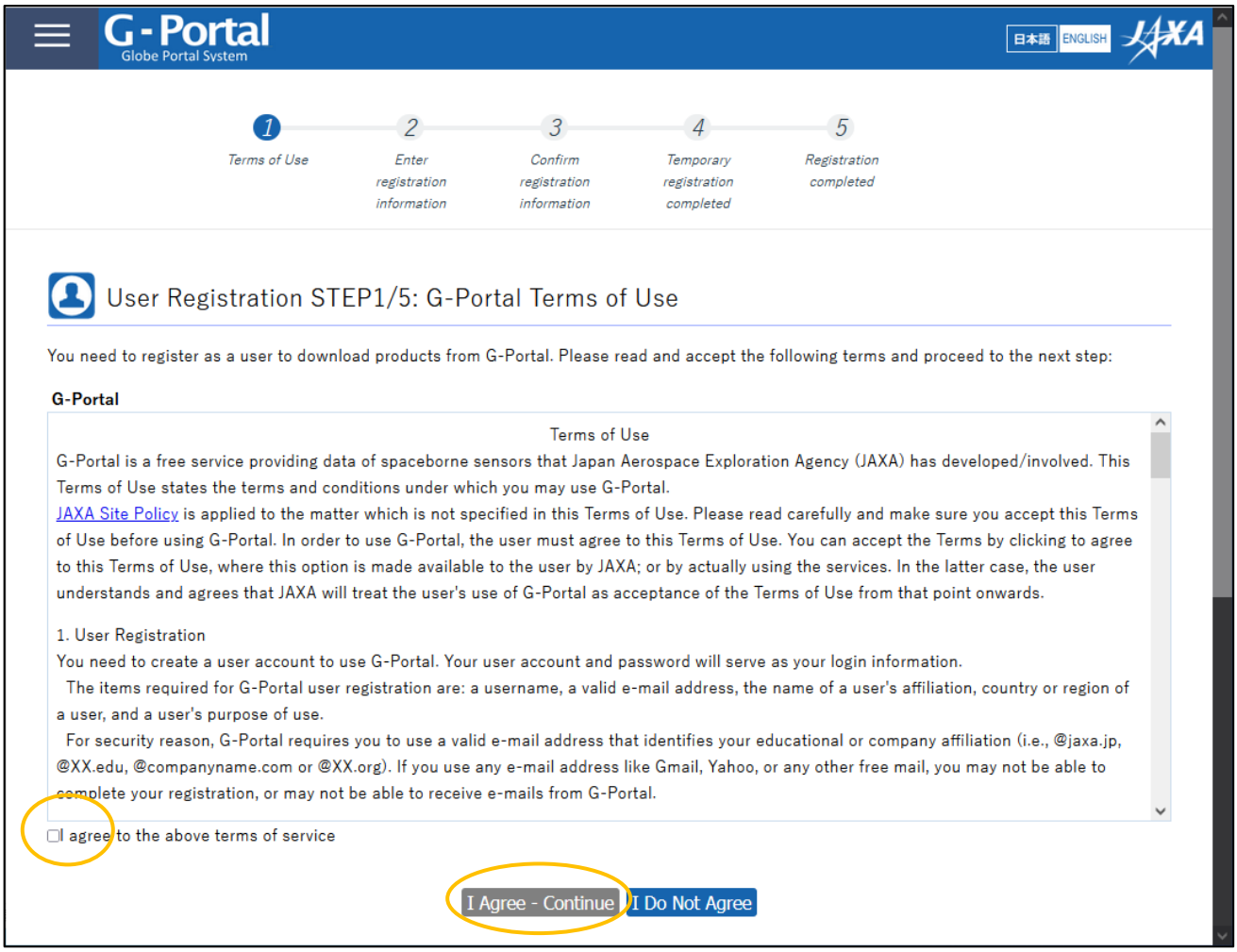
○ : Operation is confirmed. －: Operation is unconfirmed.

2. how to obtain GPM/TRMM data

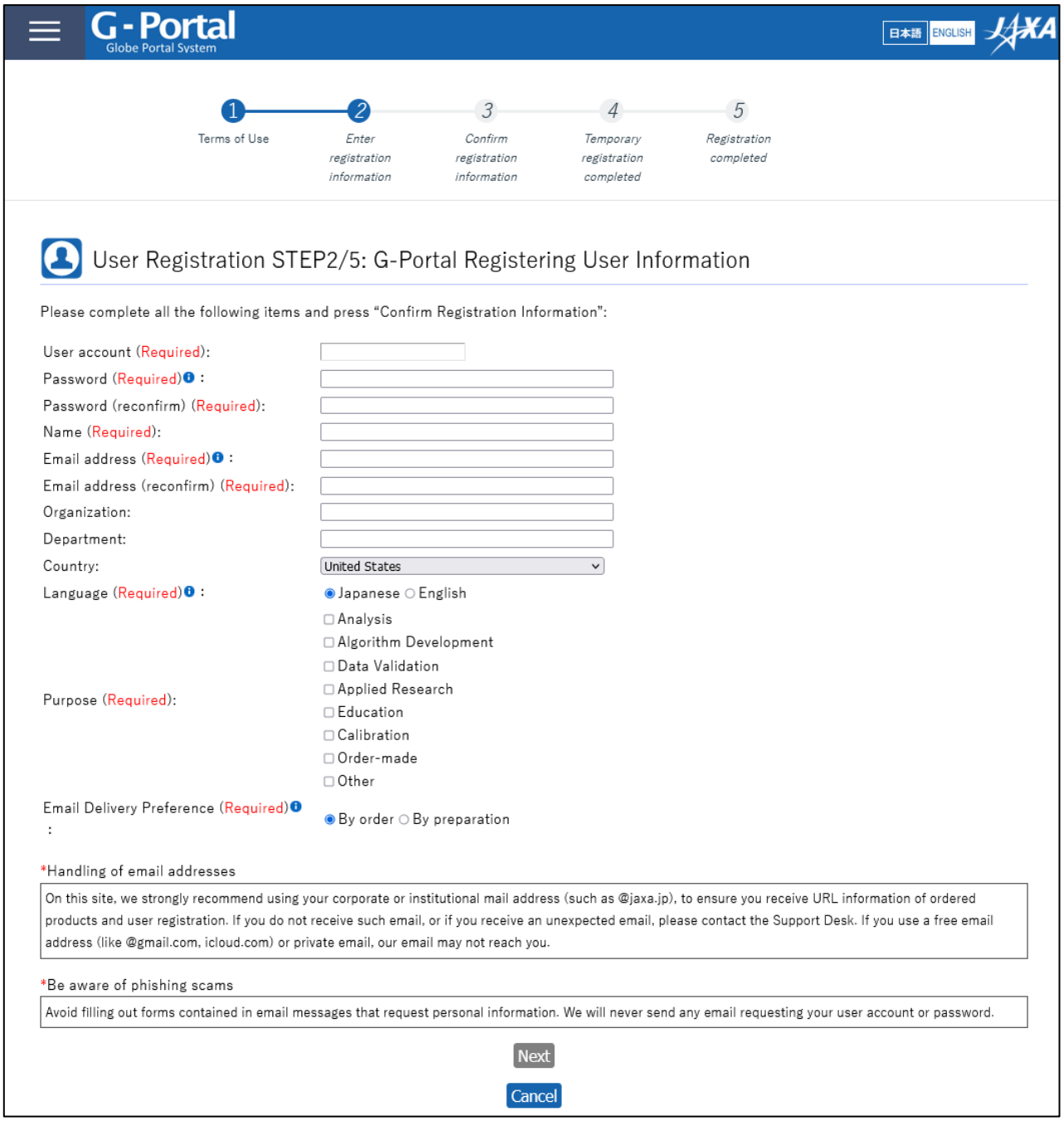
GPM/TRMM data can be obtained from the G-Portal site (https://www.gportal.jaxa.jp/gp/top.html). User registration is required to obtain the data, so please register by selecting "User Registration" from the menu on the G-Portal site.



Read the terms and conditions and click "Agree and Next.



You will be taken to the user registration screen.



For the subsequent procedures and how to obtain data after user registration, please refer to "5.2 How to Use the Data Providing Service" in the "GPM Data Users Handbook". For information on how to obtain the "GPM Data Users Handbook," please refer to "3.

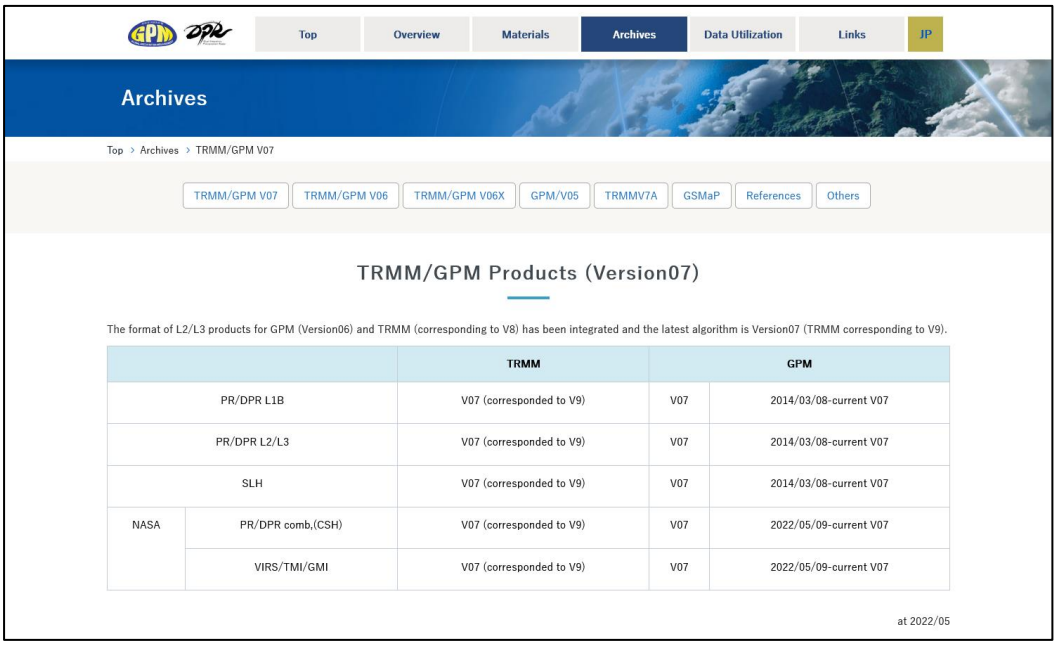
3. how to obtain related documents and sample programs

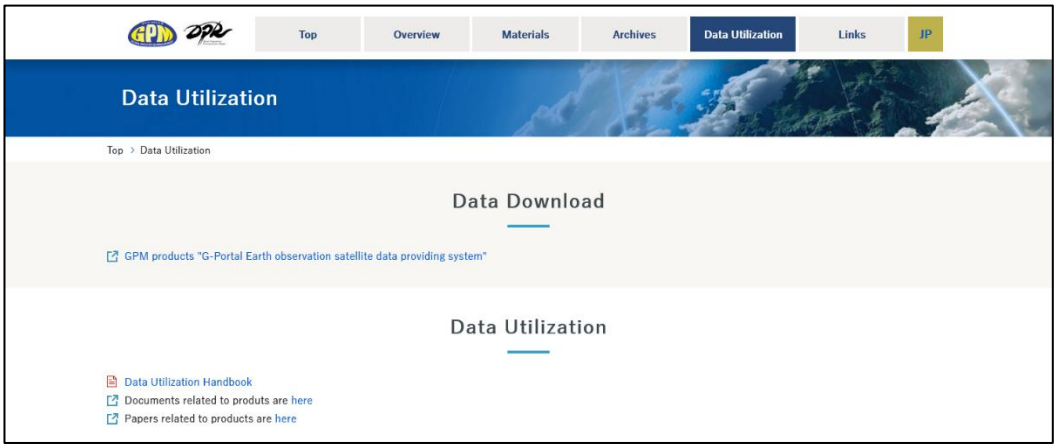
There are two types of documents related to GPM/TRMM data: documents related to data use and documents related to products. Both documents can be downloaded from the Global Precipitation Measurement Project (GPM) website (https://www.eorc.jaxa.jp/GPM/index.html). You can also download the sample codes described in this document from Top Page > Data Utilization

Documentation for GPM data use includes

GPM Data Application Handbook

file naming convention





Click "TRMM/GPM V07" to see the list of documents for product version 07.

The products, programs, and sample data described in this document are as follows

Table 3.1 List of Sample Programs

product	sample program	sample data
L1Ku	sample_L1_Ku_F.f90	GPMCOR_KUR_2112070007_0140_044170_1BS_DUB_07A.h5
L2DPR	sample_L2_DPR_F.f90	GPMCOR_DPR_2112070007_0140_044170_L2S_DD2_07A.h5
	sample_HDF5_L2_DPR_F.f90	
	sample_h5dump_L2_F.f90	
L3DPR	sample_L3_DPR_F.f90	GPMCOR_DPR_1806_M_D3M_07X.EORC.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_KUR_0901311508_1640_063883_1BS_PU1_V07X_20210630.h5
L2PR	sample_L2_PR_F.f90	GPMTRM_KUR_0901311508_1640_063883_L2S_PU2_V07X_20211129.h5
L3PR	sample_L3_PR_F.f90	3A-MO.TRMM.PR.V9-20210117.19980101-S000000-E235959.01.TRMM700.HDF5
L2SLP	sample_L2_SLP_F.f90	GPMCOR_KUR_1512282046_2218_010412_L2S_SLP_07X_20211018.EORC.h5
L3SLM	sample_L3_SLM_F.f90	GPMCOR_KUR_1512_L3S_SLM_07X_20211019.EORC.h5
L3SLG	sample_L3_SLG_F.f90	GPMCOR_DPR_2112010038_0211_044077_L3S_SLG_07A.h5
L2LHP	sample_L2_LHP_F.f90	GPMTRM_KUR_1503312350_0121_098980_L2S_LHP_07X_20211018.EORC.h5
L3LHM	sample_L3_LHM_F.f90	GPMTRM_KUR_1503_L3S_LHM_07X_20211019.EORC.h5
L3LHG	sample_L3_LHG_F.f90	GPMTRM_KUR_1503312350_0121_098980_L3S_LHG_07X_20211028.EORC.h5
GSMaP	sample_GSMaP_HDF5_F.f90	GPMMRG_MAP_2112010000_H_L3S_MCH_05A.h5
	sample_HDF5_GSMaP_F.f90	

4. installation of library tools

There are three different ways to read GPM data in FORTRAN, as shown in Table 4.1, and some methods require tools to be installed. This manual describes how to create programs for each of these methods.

Table 4.1 GPM data loading method

	How to load GPM data	Required libraries, tools	remarks
1	PPS Toolkit(TKIO)	HDF5, PPS TKIO	
2	HDF5 Library	HDF5	
3	h5dump	HDF5	

When using the PPS Toolkit (TKIO), Table 4.2 shows the relationship between the GPM data product version and the corresponding PPS Toolkit (TKIO) version.

Table 4.2 Product Version and PPS Toolkit (TKIO) Supported Versions

product	Product Version	PPS Toolkit Version	remarks
L1Ku	07	3.97.18	
L2DPR	07	3.97.18	
L3DPR	07	3.97.18	
L2GMI	05	3.80.26	
L3GMI	05	3.80.26	
L2CMB	06	3.90.0	
L3CMB	06	3.90.0	
L1PR	07	3.97.18	
L2PR	07	3.97.18	
L3PR	07	3.97.18	
L2SLP	07	3.97.18	
L3SLM	07	3.97.18	
L3SLG	07	3.97.18	
L2LHP	07	3.97.18	
L3LHM	07	3.97.18	
L3LHG	07	3.97.18	
GSMaP	04	3.80.10	

Note: PPS Toolkit (TKIO) is basically upward compatible, but may not load properly in some cases.
In that case, please refer to "5.9 About the version of PPS Toolkit (TKIO)".

The sample programs in this document have been tested in the following environments

Table 4.2 Operating Environment

(data) item	environment
calculator	Intel(R) Xeon(R) CPU ES-2665 2.4GHz
OS	Red Hat Enterprise Linux Server release 6.4
FORTRAN compiler	ifort 14.0.1
HDF5	Hdf5-1.8.9
PPS TKIO	tkio-x.xx.x

4.1 Installation of HFD5

4.1.1 Download

Download the compressed file of the source installation version of HDF5 from The HDF Group homepage (http://www.hdfgroup.org/).

*The following description assumes you have downloaded hdf5-1.8.9.tar.gz.

4.1.2 Decompression

Extract the compressed file in an appropriate working directory. You can use the following command to decompress the file.

$ tar -xzvf hdf5-1.8.9.tar.gz

After unzipping, a directory such as hdf5-1.8.9 will be created, so move to that directory.

$ cd hdf5-1.8.9

4.1.3 Compilation and Installation

Execute the following commands in order to compile and install

--prfix= specifies the directory to install in.

*In this example, the version of hdf5 is 1.8.9, so hdf5_1.8.9 is used.

Replace the version letter part with the version actually used.

<If you do not use the FORTRAN library in HDF5

$. /configure --disable-shared --prefix=/home/user1/util/hdf5_1.8.9

--with-szlib=/home/user1/util/szip_2.1

$ make

$ make install

<If you use the FORTRAN library in HDF5

$. /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 Installation of PPS Toolkit (TKIO)

PPS Toolkit (TKIO) is a library used to create programs that read GPM's HDF5 files. h5dump is not required to install when reading using the HDF5 library or h5dump.

4.2.1 Download

Download the appropriate compressed file for your environment from the following URL
https://gpmweb2https.pps.eosdis.nasa.gov/pub/PPStoolkit/GPM/

4.2.2 Decompression

Create an appropriate working directory, move the downloaded files into it, and extract the compressed files.
You can decompress it with the following command

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

4.2.3 Checking Assumptions

Refer to the tkioINSTALL.txt file in the docs directory and check the prerequisites for the downloaded PPS Toolkit (TKIO) to work. If the required libraries are not installed or the version is old, install them.

Installation of libxml2 library

```
Check docs/tkioINSTALL.txt for the version you need!
. /configure --prefix=[install DIR].
make
make install
```

Installation of zlib library

```
. /configure --prefix=[install DIR].
make
make install
```

Installation of jpeg library

```
. /configure --prefix=[install DIR] --enable-shared
make
make install
make install-lib
```

Installation of hdf4 library

```
. /configure --prefix=[install DIR] --with-zlib=[zlib install DIR] --with-jpeg=[jpeg install DIR] --with-szlib=[szlib install DIR] CC=icc F77=ifort CXX=icpc
```

```
        make
      make install
```

Installation of hdf5 library

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

4.2.4 Editing the Preferences File

Create a file to define environment variables. An example is shown below. Define environment variables that suit your environment.

```
 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 Reading the Preferences File

The following command reads the preferences file.

$ source Preferences file name

4.2.6 Compilation

Compile with the following command.

$. /INSTALL.pl compileJAVA

$. /INSTALL.pl buildRW

$. /INSTALL.pl compileRW

5.GPM data reading with PPS Toolkit (TKIO)

This section describes how to create a Fortran program using PPS Toolkit(TKIO).PPS Toolkit(TKIO) must be installed beforehand when using PPS Toolkit(TKIO).

Also, when creating a program using the PPS Toolkit (TKIO), it is necessary to know the algorithm ID beforehand. Algorithm ID is an ID for each product (data type) and is stored in the file header of the HDF5 file, and you can check the information in the file header with PPS Viewer THOR.

Table 5.1 Correspondence between product, algorithm ID, and TKIO header file

level	product	algorithm ID	TKIO header file	remarks
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	
	L2SLH	2HSLH	TK_2HSLH.h	
	L2LHP	2HSLHT	TK_2HSLHT.h	
3	L3DPR	3D PR	TK_3DPR.h	
	L3GMI	3GPROF	TK_3GPROF.h	
	L3CMB	3CMB	TK_3CMB.h	
	L3PR	3PR	TK_3PR.h	
	L3HSLH	3HSLH	TK_3HSLH.h	
	L3GSLH	3GSLH	TK_3GSLH.h	
	L3HSLHT	3HSLHT	TK_3HSLHT.h	
	L3GSLHT	3GSLHT	TK_3GSLHT.h	
	GSMaP	3GSMAPH	TK_3GSMAP.h	

When creating a program, the area to be stored must be defined according to the data to be read. GPM/TRMM data consists of dimensions named scan, angle bin, and range bin. For the relationship between scan, angle bin, and range bin, refer to "1.2 Scene Definition" in the "GPM/TRMM Data Read-in Program Guide (Appendix). For information on the composition of the data to be read, download the "GPM/DPR TRMM/PR L1 Product Format Description" and "GPM/DPR TRMM/PR L2/L3 Product Format Description" from the websites listed in Section 3, "How to Obtain Related Documents and Sample Programs. Please refer to the following website for details.

An example of creating a data loading program is shown in the next section.
Program descriptions are color-coded as follows

Explanations in red describe sample programs.

Explanations in blue describe the PPS Toolkit or satellite fundamentals.

5.1 L1 data reading

5.1.1 Source Programs

The following is an example of a program that reads L1Ku. The job name and the HDF5 file name of L1Ku are used as arguments, and the following data are read from the file specified as the argument: date and time information, latitude and longitude information, echoPower, and noisePower.

```
 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_FS/ 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:
```

Include the header file of the corresponding algorithm ID (refer to Table 5.1 Correspondence between product, algorithm ID, and TKIO header file). L1Ku includes "TK_1BKu.h" because the algorithm ID is "1BKu".

Always include in the case of Fortran.

granuleHandle1BKu is a structure that stores information on HDF5 files and is used when using the

L1BKu is the structure of the HDF5 file; it stores data for one scan.

Open HDF5 file
file:HDF5 file name (string specified in argument)
1BKu:Algorithm ID
TKREAD:Read designation
HDF5: Format type
jobname: job name (string specified by argument), granuleHandle1BKu: file pointer (specifies TKINFO structure)
1: Internal compression of files (specify 1)

Metadata read (scan count readout)
granuleHandle1BKu: file pointer (specifies TKINFO structure)
"SwathHeader": The name of the item in the HDF5 file that contains the data to be read. Information is stored. It is necessary to check the item names beforehand in the "L1 Product Format Description" or PPS Viewer THOR.
NumberScansGranule": specifies the number of scans.
numOfScan: Variable that stores the number of scans read

```
32:    do iscan=1,numOfScan
33:    status = TKreadScan( granuleHandle1BKu, L1BKu )
34:
35:    ! ScanTime Read
36:    year = L1BKu.ScanTime.
37:    month = L1BKu.ScanTime.
38:    dayOfMon = L1BKu.ScanTime.DayOfMonth
39:    hour = L1BKu.ScanTime.
40:    min = L1BKu.ScanTime.Minute
41:    sec = L1BKu.ScanTime.Second
42:    msec = L1BKu.ScanTime.MilliSecond
43:    dayOfYear = L1BKu.ScanTime.
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
```

Loop for number of scans

Scan data loading
granuleHandle1BKu: file pointer (specifies TKINFO structure)
L1BKu: Area to store read data

Read scan date and time

Latitude and longitude

Reading noisePower

echoPower loading

To verify that it is reading correctly, a portion of the data (in this case, the scan is the 3947th angle 20, data in range bin 260) is output.

Close HDF files
granuleHandle1BKu: file pointer (specifies TKINFO structure)

5.1.2 Compilation Method

The following is an example of a makefile to be used at compile time.

```
 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= $(MAIN).o
 8:
 9:INC = -I$(HDF5_INC) ¥
10:    -I$(TKIO)/inc/fortcode
11:    -I$(SZIP_INC)
12:
13:LIB = -L$(HDF5_LIB) ¥1}LIB = -L$(HDF5_LIB)
14:    -L$(TKIO)/lib ¥c
15:    -L$(SZIP_LIB)
16:
17:LIBES = -ltkcselect ¥
18:    -ltkchdf5algs -ltkchdf5 ¥c
19:    -ltkc -lm -lmfhdf -ldf -lhdf5_hl -lhdf5 -ljpeg -lsz -lz -lxml2 -ltirpc
20:
21:$(MAIN): $(OBJS)
22:    $(F90) $(FFLAGS) -o $(MAIN) $(OBJS) $(INC) $(LIB) $(LIBES)
23:
24:$(MAIN).o: $(MAIN).f90
25:    $(F90) $(FFLAGS) -c $(MAIN).f90 $(INC) $(LIB) $(LIBES)
26:clean:
27:    rm -f *.o $(MAIN)
```

The name of the program in 5.1.1.

The path to the HDF5 include and library directories.

5.1.3 Execution results

The following are the results of executing the program described in 5.1.1.

The first argument "cc" is the job name. (Any string is acceptable.)

The second argument is the HDF5 file name.

```
$ . /sample_L1_Ku_F ”cc” ”GPMCOR_KUR_2112070007_0140_044170_1BS_DUB_07A.h5”
 scan= 3946 ,angle= 19
 lat= 64.8784179687500 lon= 121.662895202637
 echoPower(260)= -29999
 noisePower= -11092
```

5.2 L2 data reading

5.2.1 Source Programs

The following is an example of a program that reads L2DPR. It takes a job name and the HDF5 file name of L2DPR as arguments, and reads the date and time information, latitude and longitude information, precipRateESurface, and precipWater data from the file specified as the argument.

```
 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 precipRateESurface(NANGLE)
17: REAL*4 precipWater(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(                                                    )
29:    ENDIF
30:




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

Include the header file of the corresponding algorithm ID (see Table 5.1 Correspondence between product, algorithm ID and TKIO header file). Include "TK_2ADPR.h" because the algorithm ID of L2DPR is "2ADPR".

Always include in the case of Fortran.

The granuleHandleL2DPR is a structure that stores information on HDF5 files and is used when using the

L2ADPR is the structure of the HDF5 file, which stores data for one scan.

Open HDF5 file
file:HDF5 file name (string specified in argument)
2ADPR: Algorithm ID
TKREAD:Read designation
HDF5: Format type
jobname:job name (string specified by argument)
granuleHandleL2DPR: file pointer (specifies

Metadata read (scan count readout)
granuleHandle2ADPR: file pointer (specifies TKINFO structure)
FS_SwathHeader": Item name in the HDF5 file for the data to be read.
Information is stored. It is necessary to check the item names beforehand in the "L1 Product Format Description" or PPS Viewer THOR.
NumberScansGranule": specifies the number of scans.

```
33:    do iscan=1,numOfScan
34:    status = TKreadScan( granuleHandleL2DPR, L2ADPR )
35:
36:    ! ScanTime Read
37: year      = L2ADPR.FS.ScanTime.
38: month     = L2ADPR.FS.ScanTime.
39:    dayOfMon = L2ADPR.FS.ScanTime.DayOfMonth
40: hour      = L2ADPR.FS.ScanTime.
41:    min = L2ADPR.FS.ScanTime.Minute
42:    sec = L2ADPR.FS.ScanTime.Second
43:    msec = L2ADPR.FS.ScanTime.MilliSecond
44: dayOfYear = L2ADPR.FS.ScanTime.
45:    secOfDay = L2ADPR.FS.ScanTime.SecondOfDay
46:
47:    do iangle=1,NANGLE
48:
49: ! Latitude and Longitude Read
50: lat(iangle) = L2ADPR.FS.Latitude(iangle)
51: lon(iangle) = L2ADPR.FS.Longitude(iangle)
52:
53: ! precipRateESurface Read
54: precipRateESurface(iangle) = L2ADPR.FS.SLV.precipRateESurface(iangle)
55:

56:    do irange=1,NRANGE
57: precipWater(irange, iangle) = L2ADPR.FS.SLV. precipWater (irange, iangle)
58: enddo
59:



60: ! print the value
61: IF(iscan .eq. 1492 .and. iangle .eq. 44 ) THEN
62: write(*,*) "scan=",iscan-1,",angle=",iangle-1
63: write(*,*) "lat=", lat(iangle), "lon=", lon(iangle)
64: write(*,*) "precipRateESurface=",precipRateESurface (iangle), "(mm/hr)"
65: write(*,*) "precipWater(101,44)=", precipWater(101, iangle), "(g/m~3)"
65: ENDIF
66: enddo
67: enddo
68:
69:    status = TKclose( granuleHandleL2DPR )
70:    STOP
71:END
```

Loop for number of scans

Scan data loading
granuleHandleL2DPR: file pointer (specifies TKINFO structure)
L2ADPR: Area to store read data

Read scan date and time

Latitude and longitude

precipRateESurface loading

precipWater loading

To check that it is read correctly, a portion of the data (in this case, the scan is the 1492nd angle 44 data) is output.

5.2.2 Compilation Method

The following is an example of a makefile to be used at compile time.

```
 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= $(MAIN).o
 8:
 9:INC = -I$(HDF5_INC) ¥
10:    -I$(TKIO)/inc/fortcode
11:    -I$(SZIP_INC)
12:
13:LIB = -L$(HDF5_LIB) ¥1}LIB = -L$(HDF5_LIB)
14:    -L$(TKIO)/lib ¥c
15:    -L$(SZIP_LIB)
16:
17:LIBES = -ltkcselect ¥
18:    -ltkchdf5algs -ltkchdf5 ¥c
19:    -ltkc -lm -lmfhdf -ldf -lhdf5_hl -lhdf5 -ljpeg -lsz -lz -lxml2 -ltirpc
20:
21:$(MAIN): $(OBJS)
22:    $(F90) $(FFLAGS) -o $(MAIN) $(OBJS) $(INC) $(LIB) $(LIBES)
23:
24:$(MAIN).o: $(MAIN).f90
25:    $(F90) $(FFLAGS) -c $(MAIN).f90 $(INC) $(LIB) $(LIBES)
26:clean:
27:    rm -f *.o $(MAIN)
```

The name of the program in 5.2.1.

5.2.3 Execution results

The following are the results of executing the program described in 5.2.1.

The first argument "bb" is the job name. (Any string is acceptable.)

The second argument is the HDF5 file name.

```
$ . /sample_L2_DPR_F ”bb” ”GPMCOR_DPR_2112070007_0140_044170_L2S_DD2_07A.h5”
scan= 1491 ,angle= 43
 lat= -19.8116989135742 lon= 30.2709083557129
 precipRateESurface= 3.003061 (mm/hr)
 precipWater(101,44)= 3.9506834E-02 (g/m^3)
```

5.3 L3 data reading

5.3.1 Source Programs

The following is an example of an L3DPR read program. It takes a job name and the name of the L3DPR HDF5 file as arguments and reads the data named precipRateESurface.mean from the file specified as the argument.

```
 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_FS/ L3DPR
 9:
10:    INTEGER status
11:    INTEGER st, rt, chn, lnL, ltL
12:    CHARACTER*255 jobname, file
13: REAL*4 precipEsurf(28,72,3,3,3,3)
14: REAL*4 lat, lon
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, 3
30: do rt=1, 3
31: do st=1, 3
32: ! precipRateESurface.mean Read
33: precipEsurf(ltL, lnL, chn, rt, st) = L3DPR.G1.precipR
chn, rt, st)
34:
35: ! print the value
36: IF(lnL .eq. 64 .and. ltL .eq. 15 .and. chn .eq. 1 .and. st .eq. 1) THEN
37: write(*,*) "st=",st-1, "rt=",rt-1
38: lat = (140.0/28.0) * (ltL-1) - 70.0 + (140.0/28.0/2)
39: lon = (360.0/72.0) * (lnL-1) - 180.0 + (360.0/72.0/2)
40: write(*,*) "lat=",lat, "lon=",lon
41: write(*,*) "chn=0 lnL=63 ltL=14 precipRateEsurface.mean=",precipEsurf(ltL, lnL,
chn, rt, st), "(mm/hr)"
42: ENDIF
```

Include the header file of the corresponding algorithm ID (see Table 5.1 Correspondence between product, algorithm ID and TKIO header file). TK_3DPR.h" is included because the algorithm ID of L3DPR is "3DPR".

Always include in the case of Fortran.

granuleHandle3DPR is a structure that stores information on HDF5 files and is used when using the

L3DPR is the structure of the HDF5 file. It stores grid data.

Open HDF5 file
file:HDF5 file name (string specified in argument)
3DPR: Algorithm ID, TKREAD: Read specification
HDF5: Format type
jobname:job name (string specified by argument)
granuleHandle3DPR: file pointer (specifies TKINFO structure)
1: Internal compression of files (specify 1)

Grid data loading
granuleHandle3DPR: file pointer (specifies TKINFO structure)
L3DPR: Area to store read data

precipRateESurface.mean loading

A portion of the data is output to confirm that it is read correctly.

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

Close HDF files
granuleHandle3DPR: file pointer (specifies TKINFO structure)

5.3.2 Compilation Method

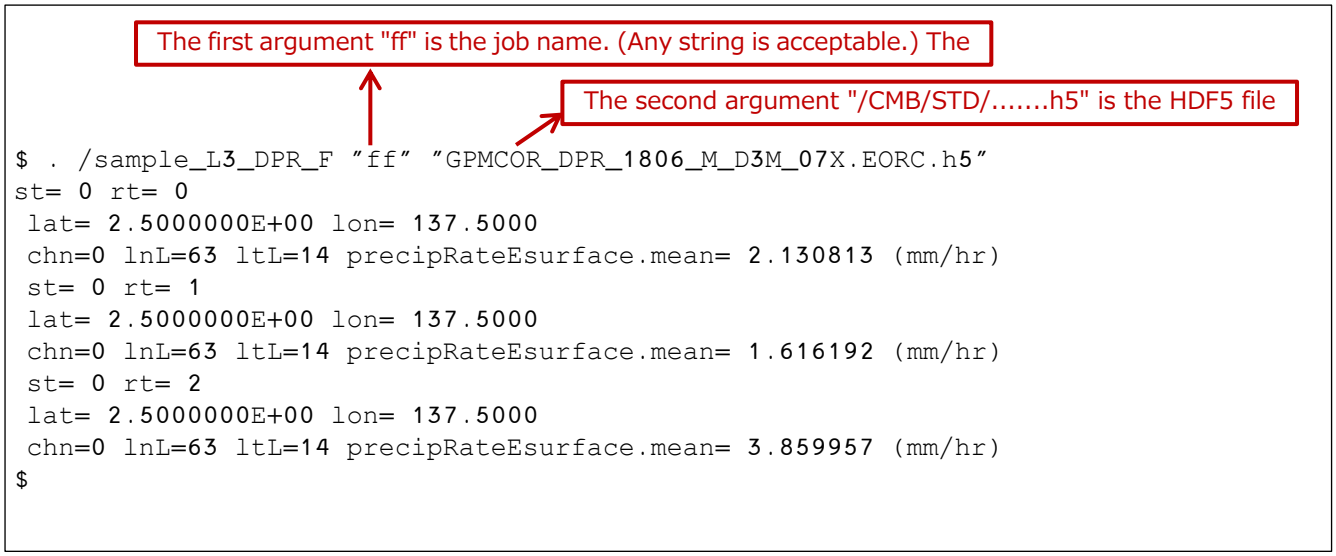
The following is an example of a makefile to be used at compile time.

```
 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= $(MAIN).o
 8:
 9:INC = -I$(HDF5_INC) ¥
10:    -I$(TKIO)/inc/fortcode
11:    -I$(SZIP_INC)
12:
13:LIB = -L$(HDF5_LIB) ¥1}LIB = -L$(HDF5_LIB)
14:    -L$(TKIO)/lib ¥c
15:    -L$(SZIP_LIB)
16:
17:LIBES = -ltkcselect ¥
18:    -ltkchdf5algs -ltkchdf5 ¥c
19:    -ltkc -lm -lmfhdf -ldf -lhdf5_hl -lhdf5 -ljpeg -lsz -lz -lxml2 -ltirpc
20:
21:$(MAIN): $(OBJS)
22:    $(F90) $(FFLAGS) -o $(MAIN) $(OBJS) $(INC) $(LIB) $(LIBES)
23:
24:$(MAIN).o: $(MAIN).f90
25:    $(F90) $(FFLAGS) -c $(MAIN).f90 $(INC) $(LIB) $(LIBES)
26:clean:
27:    rm -f *.o $(MAIN)
```

The name of the program in 5.5.1.

5.3.3 Execution results

The following are the results of executing the program described in 5.3.1.




```
$ . /sample_L3_DPR_F ”ff” ”GPMCOR_DPR_1806_M_D3M_07X.EORC.h5”
st= 0 rt= 0
 lat= 2.5000000E+00 lon= 137.5000
 chn=0 lnL=63 ltL=14 precipRateEsurface.mean= 2.130813 (mm/hr)
 st= 0 rt= 1
 lat= 2.5000000E+00 lon= 137.5000
 chn=0 lnL=63 ltL=14 precipRateEsurface.mean= 1.616192 (mm/hr)
 st= 0 rt= 2
 lat= 2.5000000E+00 lon= 137.5000
 chn=0 lnL=63 ltL=14 precipRateEsurface.mean= 3.859957 (mm/hr)
$
```

5.4 Reading GSMaP_HDF5 data

5.4.1 Source Programs

The following sample program takes a job name and the name of a GSMaP HDF5 file as arguments and reads the data named hourlyPrecipRateGC from the file specified as the argument.

```
 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: REAL*4 lat, lon
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. 946 .and. nlon .eq. 1251 ) THEN
34: lat = (180.0/1800.0) * (nlat-1) - 90.0 + (180.0/1800/2)
35: lon = (360.0/3600.0) * (nlon-1) - 180.0 + (360.0/3600.0/2)
36: write(*,*) "lat=",lat, "lon=",lon
37: write(*,*) "hourlyPrecipRateGC=",hourlyPrecipRateGC(nlat,nlon),"(mm/hr)"
38: ENDIF
39: enddo
40: enddo
41:
42:    ! HDF file close
41:    status = TKclose( granuleHandle3GSMAPH )
42:    STOP
43:END
```

Include the header file of the corresponding algorithm ID (see Table 5.1 Correspondence between product, algorithm ID and TKIO header file). Include "TK_3GSMAPH.h" because the algorithm ID of GSMaP is "3GSMAPH".

The granuleHandle3GSMAPH is a structure that stores information on HDF5 files and is used when

L3GSMAPH is the structure of the HDF5 file. It stores grid data.

Open HDF5 file
file:HDF5 file name (string specified in argument)
3GSMAPH: Algorithm ID
TKREAD:Read designation
HDF5: Format type
jobname:job name (string specified by argument)
granuleHandle3GSMAPH: file pointer (specifies TKINFO structure)
1: Internal compression of files (specify 1)

Grid data loading
granuleHandle3GSMAPH: file pointer (specifies TKINFO structure)
L3GSMAPH: Area to store read data

hourlyPrecipRateGC read

Close HDF files
granuleHandle3GSMAPH: file pointer (specifies TKINFO structure)

5.4.2 Compilation Method

The following is an example of a makefile to be used at compile time.

```
 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= $(MAIN).o
 8:
 9:INC = -I$(HDF5_INC) ¥
10:    -I$(TKIO)/inc/fortcode
11:    -I$(SZIP_INC)
12:
13:LIB = -L$(HDF5_LIB) ¥1}LIB = -L$(HDF5_LIB)
14:    -L$(TKIO)/lib ¥c
15:    -L$(SZIP_LIB)
16:
17:LIBES = -ltkcselect ¥
18:    -ltkchdf5algs -ltkchdf5 ¥c
19:    -ltkc -lm -lmfhdf -ldf -lhdf5_hl -lhdf5 -ljpeg -lsz -lz -lxml2 -ltirpc
20:
21:$(MAIN): $(OBJS)
22:    $(F90) $(FFLAGS) -o $(MAIN) $(OBJS) $(INC) $(LIB) $(LIBES)
23:
24:$(MAIN).o: $(MAIN).f90
25:    $(F90) $(FFLAGS) -c $(MAIN).f90 $(INC) $(LIB) $(LIBES)
26:clean:
27:    rm -f *.o $(MAIN)
```

The name of the program in 5.8.1.

The path to the HDF5 include and library directories.

5.4.3 Execution results

The following are the results of executing the program described in 5.4.1.

The first argument "hh" is the job name. (Any string is acceptable.)

The second argument is the HDF5 file name.

```
$ . /sample_GSMaP_HDF5_F "hh" "GPMMRG_MAP_2112010000_H_L3S_MCH_05A.h5"
lat= 4.550000 lon= -54.95000
 hourlyPrecipRateGC= 0.6264970 (mm/hr)$
```

5.5 About the version of PPS Toolkit (TKIO)

If the version of the PPS Toolkit (TKIO) installed differs from the version in which the HDF5 file was created, it may not be read properly. In that case, you need to check the version of the HDF5 file and change the program to the header file and algorithm ID that match the version.

To find out the version of an HDF5 file, use PPS Viewer THOR to read the FileInfo of the HDF5 file and check the values of "DataFormatVersion" and "TKCodeBuildVersion".

DataFormatVersion=bk

TKCodeBuildVersion=2

the version is bk2.

The program changes are in the following three areas

1) Header file name to include
2) algorithm ID
3) Header file reference point

Changes to the header file and algorithm ID add a version notation. If the header file is "TK_2ADPR.h" and the algorithm ID is "2ADPR", the header file becomes "TK_2DPR_bk2.h" and the algorithm ID becomes "2ADPR_bk2".

As the header file is changed, the sections that refer to the contents of the header file must also be changed to match the contents of the new header file.

Below is an example of a modified L2DPR data loading sample program.

```
 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:
```

It is this file that is the header file to be included and changed according to the version. (TK_xxxx.h)
Change to "TK_2ADPR_bk2.h".

```
 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.
 38:    month = L2ADPR.NS.ScanTime.
 39:    dayOfMon = L2ADPR.NS.ScanTime.DayOfMonth
 40:    hour = L2ADPR.NS.ScanTime.
 41:    min = L2ADPR.NS.ScanTime.Minute
 42:    sec = L2ADPR.NS.ScanTime.Second
 43:    msec = L2ADPR.NS.ScanTime.MilliSecond
 44:    dayOfYear = L2ADPR.NS.ScanTime.
 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/hr)"
 65: ENDIF
 66: enddo
 67: enddo
 68:
 69:    status = TKclose( granuleHandleL2DPR )
 70:    STOP
 71:END
```

The algorithm ID is specified by the This part. Add a version. Change to "2ADPR_bk2".

6. GPM data loading with HDF library

Describes how to create a Fortran program using the HDF library.

Explanations in red describe sample programs.

Explanations in blue describe the HDF library or satellite fundamentals.

6.1 Loading L2DPR data

6.1.1 Source Programs

The following sample program reads the data named precipRateESurface from the file specified by filename.

```
 1:PROGRAM SAMPLE
 2:
 3: use hdf5 ! This module contains all necessary modules
 4:
 5: CHARACTER(LEN=34), PARAMETER :: filename = "/DPRL2/
GPMCOR_DPR_2112070007_0140_044170_L2S_DD2_07A.h5"
 6:
 7: CHARACTER(LEN=26), PARAMETER :: dsetname = "/HS/SLV/precipRateESurface"
 8:
 9: INTEGER(HID_T) :: file_id ! File identifier
10: INTEGER(HID_T) :: dset_id ! Dataset identifier
11: REAL*4, DIMENSION(49,7934) :: 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:
```

The name of the HDF5 file to be read.

The name of the data to be read from the HDF5 file.

HDF5 library initialization
error: Error information (0:normal)

HDF5 file open
filename: HDF5 file name
H5F_ACC_RDWR_F: Access designation
file_id: File ID (output information)
error: Error information (0:normal)

Open Data Sets
file_id: Specify the value output by h5fopen_f.
dsetname: Name of the data to be read.
dset_id: Data set ID (output information)
error: Error information (0:normal)

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

data loading
dset_id: Specify the value output by h5dopen_f.

```
25: data_dims(1) = 49
26: data_dims(2) = 7934
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(44,1492)=",precipRateESurface(44,1492),
"(mm/hr)"
33:
34: ! Close the dataset.
35: CALL h5dclose_f(dset_id, error)
36:
37: ! Close the file.
38: CALL h5fclose_f(file_id, error)
30:
40: ! Close FORTRAN interface.
41: CALL h5close_f(error)
42:
43:   STOP
44:END
```

6.1.2 Compilation Method

The following is an example of a makefile to be used at compile time.

```
  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) ¥1}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)
```

Name of the program in 6.1.1.

The path to the HDF5/HDF5 include and library directories.

6.1.3 Execution Results

The following are the results of executing the program described in 6.1.1.

```
$ . /sample_HDF5_L2_DPR_F
 filename= /DPRL2/GPMCOR_DPR_2112070007_0140_044170_L2S_DD2_07A.h5
dataset name=/FS/SLV/precipRateESurface
 precipRateESurface(44,1492)= 3.003061 (mm/hr)$
```

6.2 Loading L3DPR data

6.2.1 Source Programs

The following sample program reads the data named precipRateESurface from the file specified by filename.

```
  1:PROGRAM SAMPLE
  2:
  3: use hdf5 ! This module contains all necessary modules
  4:
  5: CHARACTER(LEN=38), PARAMETER :: filename = "/DPRL3/
GPMCOR_DPR_1806_M_D3M_07X.EORC.h5"
  6:
  7: CHARACTER(LEN=34), PARAMETER :: dsetname = "/FS/G1/precipRateESurface/mean"
  8:
  9: INTEGER(HID_T) :: file_id ! File identifier
 10: INTEGER(HID_T) :: dset_id ! Dataset identifier
 11: REAL*4, DIMENSION(28,72,3,3,3) :: precipRateESurface ! Data buffers
 12: REAL*4 lat, lon
 13: INTEGER(HSIZE_T), DIMENSION(5) :: data_dims
 14:  INTEGER error
 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)
```

The name of the HDF5 file to be read.

The name of the data to be read from the HDF5 file.

HDF5 library initialization
error: Error information (0:normal)

HDF5 file open
filename: HDF5 file name
H5F_ACC_RDWR_F: Access designation
file_id: File ID (output information)
error: Error information (0:normal)

Open Data Sets
file_id: Specify the value output by h5fopen_f.
dsetname: Name of the data to be read.
dset_id: Data set ID (output information)
error: Error information (0:normal)

```
 23:
 24: ! Read precipRateESurface.
 25: data_dims(1) = 28
 26: data_dims(2) = 72
 27: data_dims(3) = 3
 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: lat = (140.0/28.0) * (15-1) - 70.0 + (140.0/28.0/2)
 36: lon = (360.0/72.0) * (64-1) - 180.0 + (360.0/72.0/2)
 37: write(*,*) "lat=",lat, "lon=",lon
 38: write(*,*)
"precipRateESurface.mean(15,64,1,1,1)=",precipRateESurface(15,64,1,1,1)
 39: write(*,*)
"precipRateESurface.mean(15,64,1,2,1)=",precipRateESurface(15,64,1,2,1)
 40:
 41: ! Close the dataset.
 42: CALL h5dclose_f(dset_id, error)
 43:
 44: ! Close the file.
 45: CALL h5fclose_f(file_id, error)
 46:
 47: ! Close FORTRAN interface.
 48: CALL h5close_f(error)
 49:
 50:   STOP
 51:END
```

data loading
dset_id: Specify the value output by h5dopen_f.

6.2.2 Compilation Method

The following is an example of a makefile to be used at compile time.

```
  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) ¥1}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)
```

The name of the program in 6.2.1.

The path to the HDF5/HDF5 include and library directories.

6.2.3 Execution Results

The following are the results of executing the program described in 6.2.1.

```
$ . /sample_HDF5_L3_DPR_F
 filename= /DPRL3/GPMCOR_DPR_1806_M_D3M_07X.EORC.h5
 dataset name=/FS/G1/precipRateESurface/mean
lat= 2.500000 lon= 137.5000
 precipRateESurface.mean(15,64,1,1,1,1) = 2.130813
 precipRateESurface.mean(15,64,1,2,1)= 1.616192
```

6.3 Reading GSMaP_HDF5 data

6.3.1 Source Programs

The following sample program reads the data named `hourlyPrecipRateGC` from the file specified by filename.

```
  1:PROGRAM SAMPLE
  2:
  3: use hdf5 ! This module contains all necessary modules
  4:
  5: CHARACTER(LEN=35), PARAMETER :: filename = "/GSMaP/
GPMMRG_MAP_2112010000_H_L3S_MCH_05A.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: REAL*4 lat, lon
 13: INTEGER(HSIZE_T), DIMENSION(2) :: data_dims
 14:   INTEGER error
 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:
```

The name of the HDF5 file to be read.

The name of the data to be read from the HDF5 file.

HDF5 library initialization
error: Error information (0:normal)

HDF5 file open
filename: HDF5 file name
H5F_ACC_RDWR_F: Access designation
file_id: File ID (output information)
error: Error information (0:normal)

Open Data Sets
file_id: Specify the value output by h5fopen_f.
dsetname: Name of the data to be read.
dset_id: Data set ID (output information)
error: Error information (0:normal)

```
 29: ! print the value
 30: write(*,*) "filename=",filename,"""
 31: write(*,*) "dataset name=",dsetname,"""
 32: lat = (180.0/1800.0) * (946-1) - 90.0 + (180.0/1800.0/2)
 33: lon = (360.0/3600.0) * (1251-1) - 180.0 + (360.0/3600.0/2)
 34: write(*,*) "lat=",lat, "lon=",lon
 35: write(*,*) "hourlyPrecipRateGC(946,1251)=",hourlyPrecipRateGC(946,1251),
"(mm/hr)"
 36:
 37: ! Close the dataset.
 38: CALL h5dclose_f(dset_id, error)
 39:
 40: ! Close the file.
 41: CALL h5fclose_f(file_id, error)
 42:
 43: ! Close FORTRAN interface.
 44: CALL h5close_f(error)
 45:
 46:   STOP
 47:END
```

6.3.2 Compilation Method

The following is an example of a makefile to be used at compile time.

```
  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) ¥1}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)
```

Name of the program in 6.3.1.

6.3.3 Execution Results

The following are the results of executing the program described in 6.3.1.

```
$ . /sample_HDF5_GSMaP_F
 filename=/GSMaP/GPMMRG_MAP_2112010000_H_L3S_MCH_05A.h5
 dataset name=/Grid/hourlyPrecipRateGC
lat= 4.550000 lon= -54.95000
 hourlyPrecipRateGC(946,1251)= 0.6264970 (mm/hr)$
```

7. read GPM data with h5dump

Describes how to use h5dump to create a binary file of the data you want to read from an HDF5 file and how to create a Fortran program to read that binary file.

Explanations in red describe sample programs.

Explanations in blue describe the HDF library or satellite fundamentals.

7.1 L2 data reading

7.1.1 Creating Binary Files

Here is an example of creating a shell that uses h5dump to create a binary file.

```
 1:#! /bin/tcsh -f
 2:
 3:set h5dump_bin=/home/tool/hdf5-1.8.9/bin/h5dump
 4:set file_dir=/h5file/
 5:set OUTPUT=/binfile/
 6:
 7:cd ${file_dir}
 8:set file_in=(`ls *044170* |sed 's/.h5/ /' `)
 9:mkdir -p ${OUTPUT}
10:cd ${OUTPUT}
11:
12:foreach file ($file_in)
13:echo ${file}
14:$h5dump_bin -d FS/SLV/precipRateESurface -b -o ${file}.bin
${file_dir}/${file}.h5
15:end
```

The full path to h5dump is specified.

The directory in which the HDF5 files reside is specified.

Specifies the directory to which binary files are output.

Specify if you want to narrow down the files to be processed in the directory where HDF5 files exist. *008435*" means that only files with "008435" in the file name will be targeted.

The extension part of the file name is removed.

Loop for the number of

Binary file creation
A binary file is created from the read file with the data specified by -d and the file name specified by -b-o.

When the above shell is executed, the following is displayed and the binary file is created in the directory specified in OUTPUT.

```
$ . /dump_L2.sh
HDF5 ". /h5file/GPMCOR_DPR_2112070007_0140_044170_L2S_DD2_07A.h5" {
DATASET "FS/SLV/precipRateESurface" {
   DATATYPE H5T_IEEE_F32LE
   DATASPACE SIMPLE { ( 7932, 49 ) / ( H5S_UNLIMITED, 49 ) }
   DATA {
   }
}
}
```

7.1.2 Source Programs

The following sample program reads information from a binary file specified by inputfile.

```
  1:program sample
  2:
  3: CHARACTER(LEN=66), PARAMETER :: filename ="/binfile/
GPMCOR_DPR_2112070007_0140_044170_L2S_DD2_07A.bin"
  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(44,1492)=",precipRateESurface(44,1492),
"(mm/hr)"
 17: write(*,*) "precipRateESurface(45,1492)=",precipRateESurface(45,1492),
"(mm/hr)"
 18:
 19: ! binary file close
 20: close(10)
 21:
 22:end
```

The binary file created in 7.1.1 is specified.

Open binary file
recl: record length (specifies the size of the

Binary file loading
All data is read into precipRateESurface at one time.

7.1.3 Compile Method

The following is an example of a makefile to be used at compile time.

```
  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) ¥1}{2}{3}{4}{5}{5
 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)
```

7.1.4 Execution Results

The following are the results of executing the program described in 7.1.2.

```
$ . /sample_h5dump_L2_F
 filename=/binfile/GPMCOR_DPR_2112070007_0140_044170_L2S_DD2_07A.bin
 precipRateESurface(44,1492)= 3.003061 (mm/hr)
 precipRateESurface(45,1492)= 3.324327 (mm/hr)$
```

7.2 L3 data reading

7.2.1 Creating Binary Files

Here is an example of creating a shell that uses h5dump to create a binary file.

```
 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
```

The full path to h5dump is specified.

The directory in which the HDF5 files reside is specified.

Specifies the directory to which binary files are output.

Specify if you want to narrow down the files to be processed in the directory where HDF5 files exist. *150801*" means that only files with "150801" in the file name will be targeted.

The extension part of the file name is removed.

Loop for the number of

Binary file creation
A binary file is created from the read file with the data specified by -d and the file name specified by -b-o.

When the above shell is executed, the following is displayed and the binary file is created in the directory specified in OUTPUT.

```
$ . /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, 3, 72, 28 ) / ( 3, 3, 3, 72, 28 ) }
   DATA {
   }
}
}
$
```

7.2.2 Source Programs

The following sample program reads information from a binary file specified by inputfile.

```
  1:program sample
  2:

  3: CHARACTER(LEN=74), PARAMETER :: filename ="/binfile/
GPMCOR_DPR_1806_M_D3M_07X.EORC.precipRateESurface_mean "
  4:
  5: ! read data area
  6: real*4 precipRateESurface(28,72,3,3,3)
  7:


  8: ! binary file open
  9: open(10,file=filename,access='direct',status='old',recl=4*28*72*3*3*3)
 10:


 11: ! binary file read
 12: read(10,rec=1) precipRateESurface
 13:
 14: ! print the value
 15: write(*,*) "filename=",filename
 16: lat = (140.0/28.0) * (15-1) - 70.0 + (140.0/28.0/2)
 17: lon = (360.0/72.0) * (64-1) - 180.0 + (360.0/72.0/2)
 18: write(*,*) "lat=",lat, "lon=",lon
 19: write(*,*) "precipRateESurface.mean(15,64,1,1,1)=",
precipRateESurface(15,64,1,1,1)
 20: write(*,*) "precipRateESurface.mean(15,64,2,2,2)=",
precipRateESurface(15,64,2,2,2)
 21:
 22: ! binary file close
 23: close(10)
 24:
 25:end
```

The binary file created in 7.2.1 is specified.

Open binary file
recl: record length (specifies the size of the

Binary file loading
All data is read into precipRateESurface at one time.

7.2.3 Compilation Method

The following is an example of a makefile to be used at compile time.

```
  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) ¥1}{2}{3}{4}{5}{5}{6}{7
 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)
```

The name of the program in 7.2.1.

7.2.4 Execution Results

The following are the results of executing the program described in 7.2.2.

```
$ . /sample_h5dump_L3_F
 filename=/binfile/GPMCOR_DPR_1806_M_D3M_07X.EORC.precipRateESurface_mean
 lat= 2.500000 lon= 137.5000
 precipRateESurface.mean(15,64,1,1,1,1) = 2.130813
 precipRateESurface.mean(15,64,2,2,2) = 2.254316
```

revision history

version number	Date	Revised contents	remarks
1	2016/1/26		
2	2017/9/13	1. Introduction: python description added to Table 1.1, flowchart revised accordingly. Table 1.2 Sample code operation check table was added. 4. installation of library tools: Table 4.2 Product bar version and PPS Toolkit (TKIO). In Table 4.3 Operating Environment, where it says tkio-3.70.7 Changed to tkio-x.xx.x 4.2.4Edit configuration file: Change "tkio-3.70.7" to "tkio-x.xx.x".	
3	Mar 15, 2018	2. Related documents and sample programs available: Table 3.1 sample program list added.	
4	3/8/2019	1.-3. Correction due to addition of TRMM and renewal of GPM site Table 4.2 Product version and TKIO supported versions are modified to be listed for each product. Added a list of algorithm IDs and changed the sample programs to include one per level.	
5	12/6/2021	1. modified to GSMaP product version 5 and GPM/TRMM product version 7. 3. revised availability of related documentation and sample programs Correction of URL for TKIO download	
6	12/24/2021	3.-5. Added descriptions in Table 3.1, Table 4.2, and Table 5.1 with the addition of sample codes for SLP/SLM/SLG/LHP/LHM/LHG 5.-7. Corrected V7 changes in the program.	
7	1/21/2022	5.-7. Corrections due to modification of sample programs (review of display position, addition of lat/lon to display items, etc.)	
8	2022/2/4	Error Correction	