GPM/TRMM Data reading program guide (FORTRAN version)

2026/06/01

9th 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 08 for GPM/TRMM and with product version 05 for GSMaP.

Table of Contents

1. 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 08, 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	Using Commercial off-the-shelf (COTS) tools such as QGIS, Panoply, and HDFView	GPM/TRMM Data reading program guide (COTS version)	
3	Use GeoTIFF	GPM/TRMM Data reading program guide (GeoTIFF Conversion)	
4	Use IDL	GPM/TRMM Data Loading Program Guide (IDL version)	
5	Use C	GPM/TRMM data reading program guide (C language version)	
6	Using FORTRAN	GPM/TRMM Data Loading Program Guide (FORTRAN Edition)	
7	Using Python	GPM/TRMM data reading program guide (Python version)	

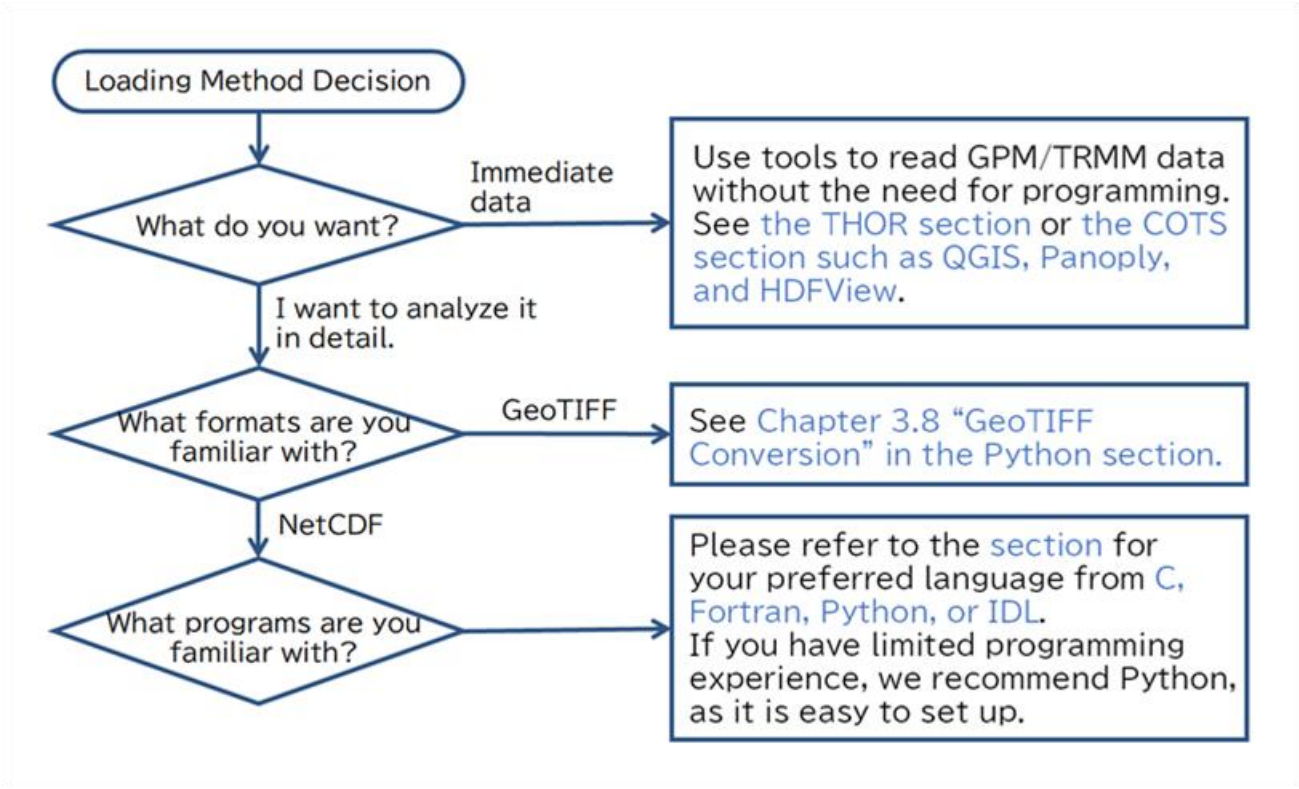


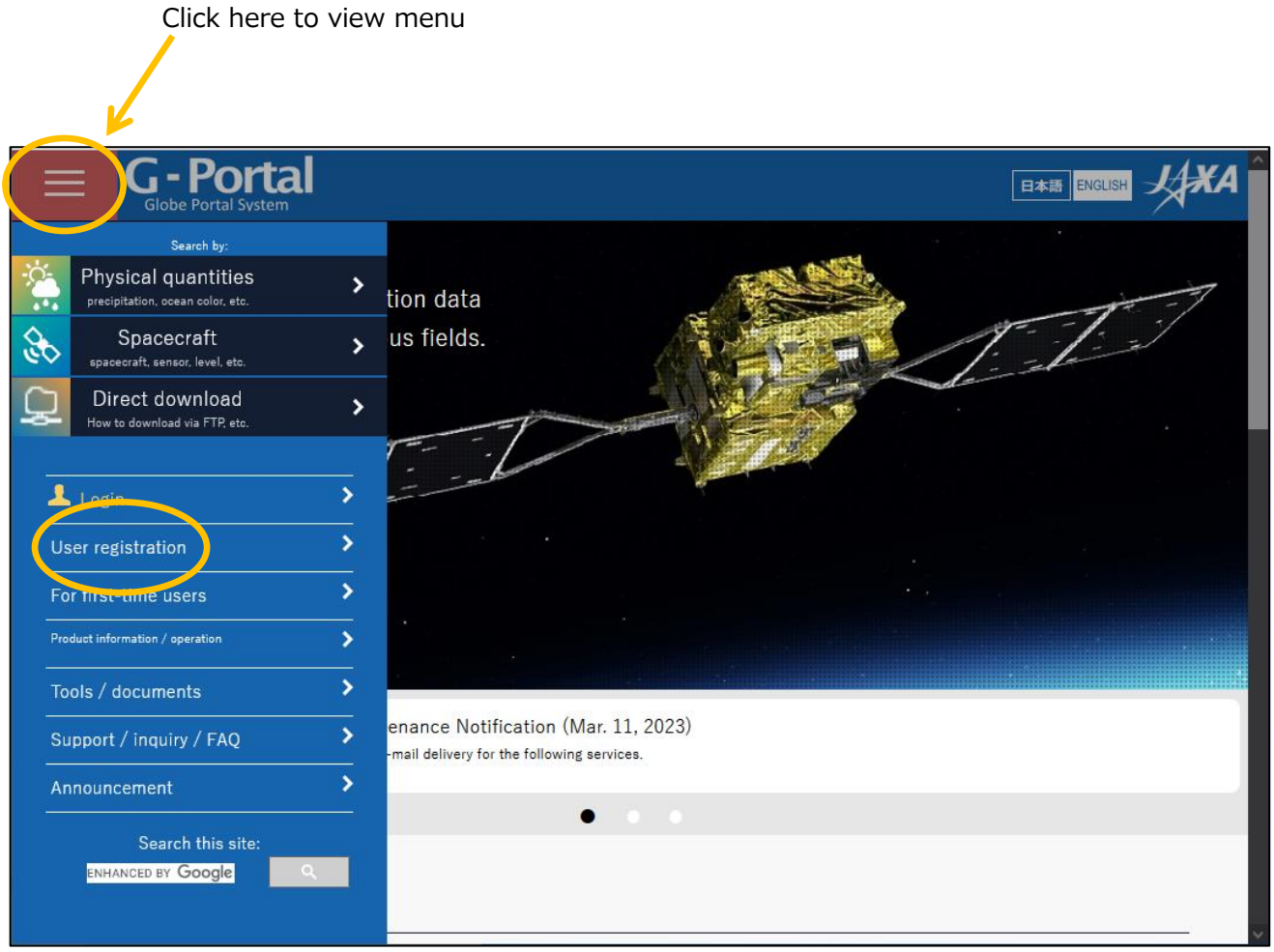
Table 1.2 Sample Program Operation Check Table

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

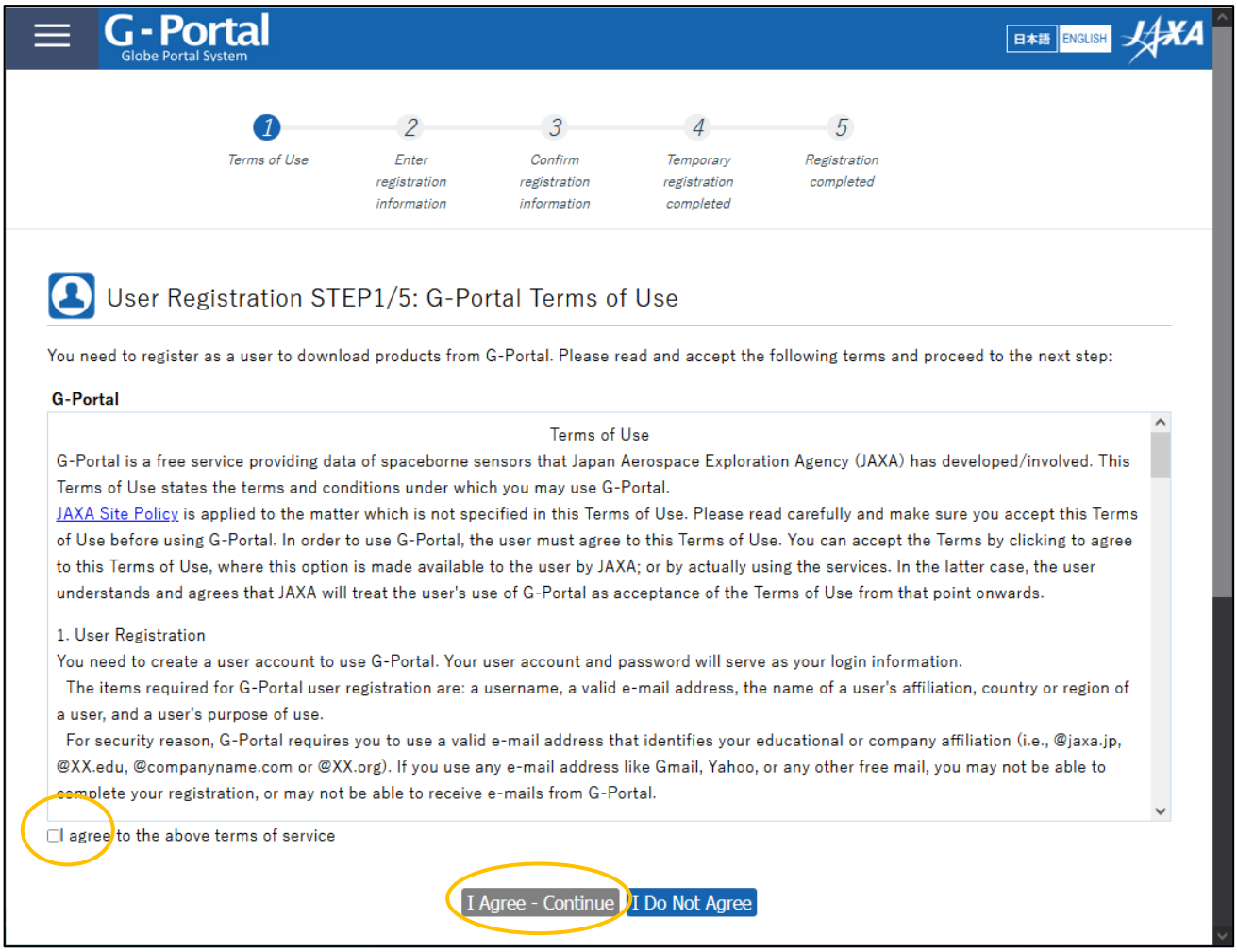
○ : Operation is confirmed. －: Operation is unconfirmed.

2. how to obtain GPM/TRMM data

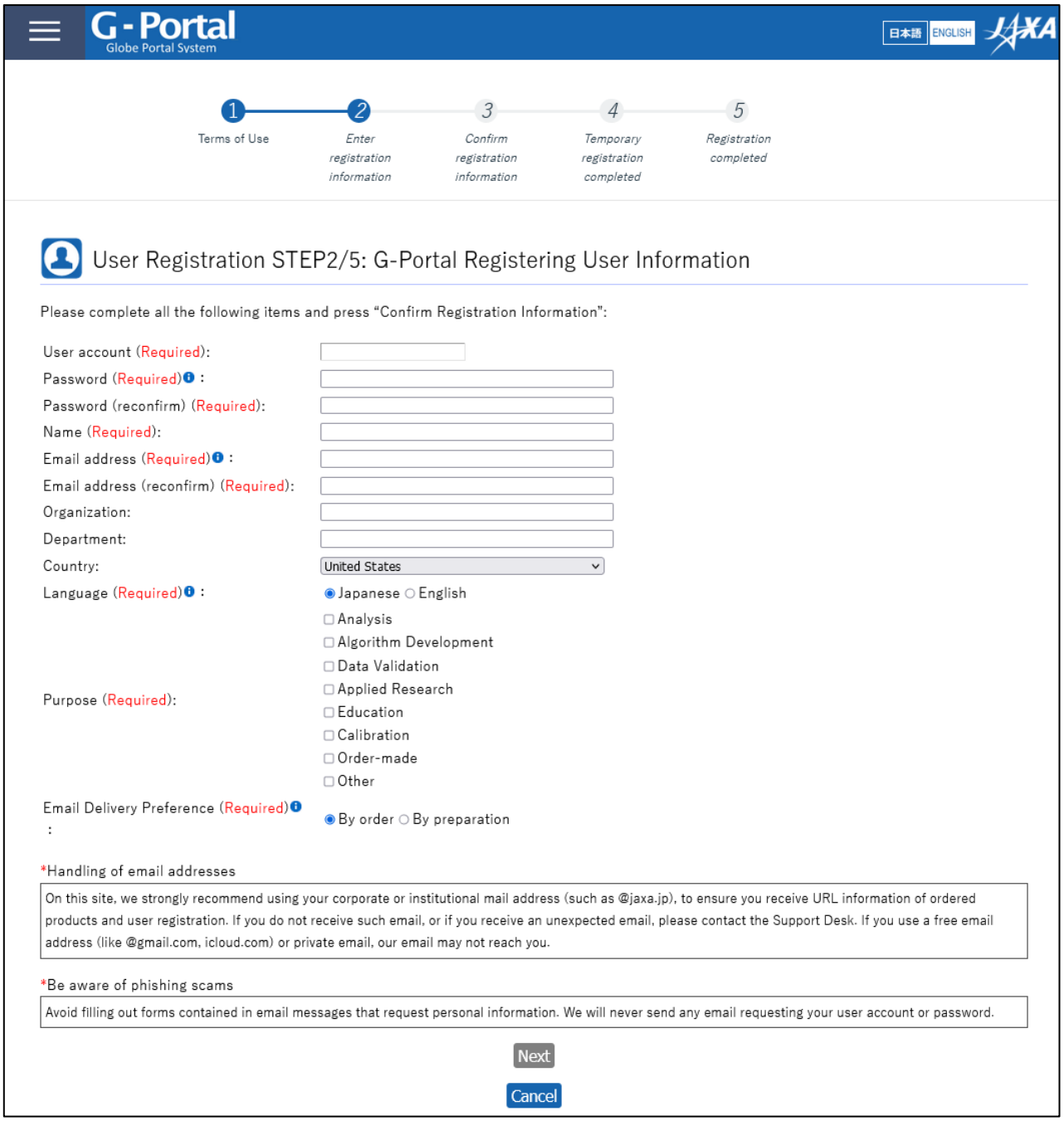
GPM/TRMM data can be obtained from the G-Portal site (https://www.gportal.jaxa.jp/gp/top.html). User registration is required to obtain the data, so please register by selecting "User Registration" from the menu on the G-Portal site.



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.

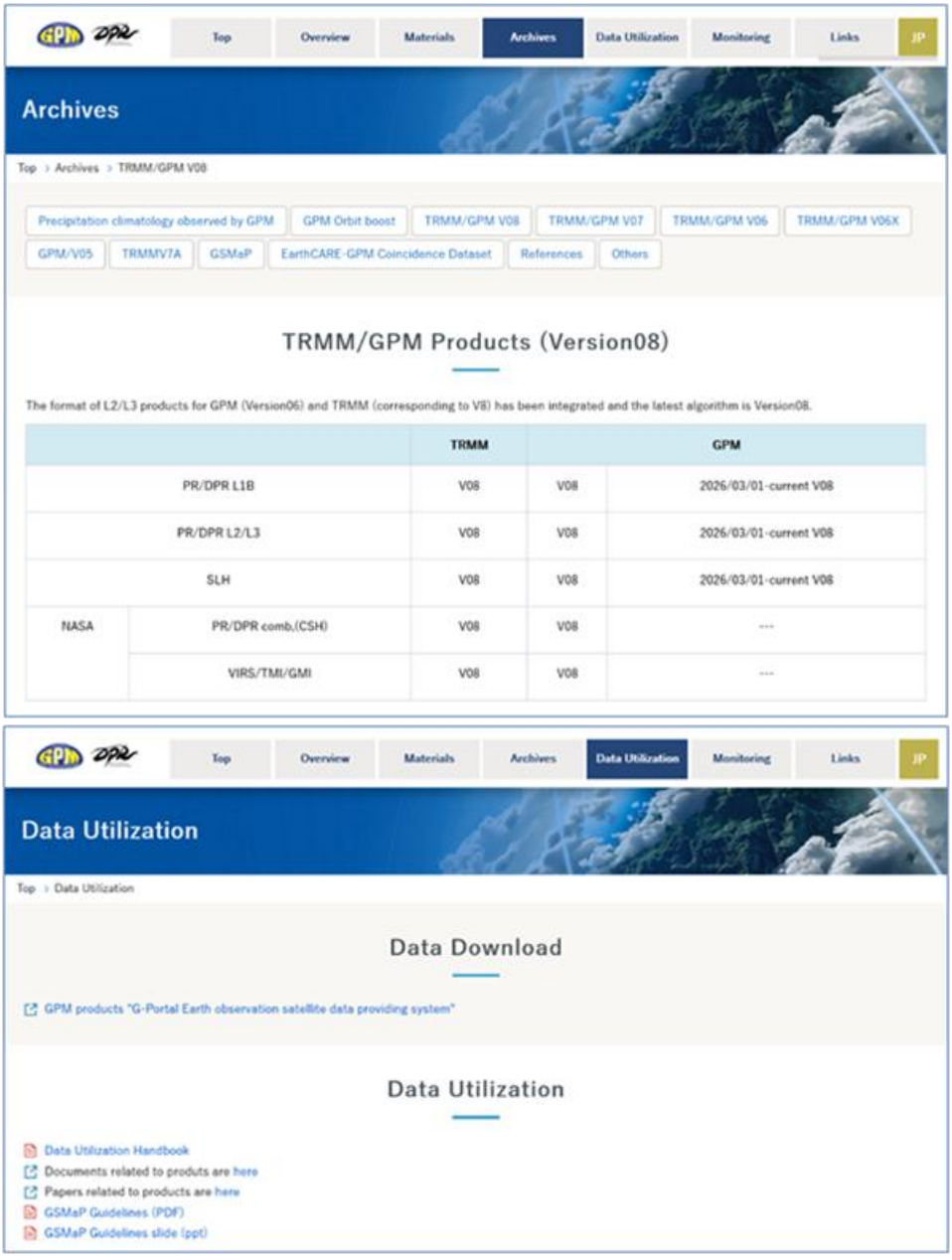
3. how to obtain related documents and sample programs

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/en/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 V08" to see the list of documents for product version 08.

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_2604010211_0345_068594_1BS_DUB_08A.nc
L2DPR	sample_L2_DPR_F.f90	GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc
	sample_HDF5_L2_DPR_F.f90	
	sample_h5dump_L2_F.f90	
L3DPR	sample_L3_DPR_F.f90	GPMCOR_DPR_2603_M_L3S_D3M_08A.nc
	sample_HDF5_L3_DPR_F.f90	
	sample_h5dump_L3_F.f90	
L2GMI	sample_L2_GMI_F.f90	GPMCOR_GMI_2604010211_0345_068594_L2S_GL2_08A.nc
L3GMI	sample_L3_GMI_F.f90	GPMCOR_GMI_2603_M_L3S_GL3_08A.nc
L2SLP	sample_L2_SLP_F.f90	GPMCOR_DPR_2604010211_0345_068594_L2S_SLP_08A.nc
L3SLM	sample_L3_SLM_F.f90	GPMCOR_DPR_2603_M_L3S_SLM_08A.nc
L3SLG	sample_L3_SLG_F.f90	GPMCOR_DPR_2604010211_0345_068594_L3S_SLG_08A.nc

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, NetCDF, PPS TKIO	See Table 4.2
2	HDF5 Library	HDF5, NetCDF	
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	08	3.102	
L2DPR	08	3.102	
L3DPR	08	3.102	
L2GMI	08	3.102	
L3GMI	08	3.102	
L2SLP	08	3.102	
L3SLM	08	3.102	
L3SLG	08	3.102	

Note: PPS Toolkit (TKIO) is basically upward compatible, but may not load properly in some cases. In that case, please refer to "5.4 About the version of PPS Toolkit (TKIO)".

The sample programs in this document have been tested in the following environments. Regarding FORTRAN compilers, we recommend using ifort if it is available.

Table 4.2 Operating Environment

(data) item	environment	remarks
calculator	Intel(R) Xeon(R) CPU E5-2665 0 @ 2.40GHz	
OS	AlmaLinux release 8.8	
FORTRAN compiler	GNU Fortran (GCC) 8.5.0 20210514	When reading using the HDF5 library
	ifx (IFX) 2025.3.2 20260112	When reading using PPS Toolkit (TKIO)
HDF5	Hdf5-1.10.8	
NetCDF	netcdf-4.9.2	
PPS TKIO	tkio-3.102	

4.1 Installation of HFD5

Since Fortran modules (include/*.mod) are required, you must build from the source code.

The compiler used for building must be the same as the one used to build the sample programs.

* While gfortran is used in the following instructions, we recommend using ifort if it is available. Please note that you cannot build using ifort's successor (ifx).

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 hdf5-1.10.8.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 hdf5-1.10.8.tar.gz

After unzipping, a directory such as hdf5-1.10.8 will be created, so move to that directory.

$ cd hdf5-1.10.8

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 1.10.8, so hdf5_1.10.8 is used.

Replace the version letter part with the version actually used.

```
$ ./configure --enable-static-exec --prefix=/home/user1/util/hdf5_1.10.8 --enable-fortran
FC=gfortran
$ make
$ make install
```

4.2 Installation of NetCDF

To link the installed HDF5 library, build it from the source code.

4.2.1 Download

Download the compressed file for the NetCDF source installation from the Unidata homepage (https://www.unidata.ucar.edu/).

*The following description assumes you have downloaded netcdf-c-4.9.2.tar.gz.

4.2.2 Decompression

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

```
$ tar -xzvf netcdf-c-4.9.2.tar.gz
```

After unzipping, a directory such as netcdf-c-4.9.2 will be created, so move to that directory.

```
$ cd netcdf-c-4.9.2
```

4.2.3 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 NetCDF is 4.9.2, so netcdf-c-4.9.2 is used.

Replace the version letter part with the version actually used.

```
$ ./configure --prefix=/home/user1/util/netcdf-c-4.9.2 --enable-fortran CC=gcc ¥
              CPPFLAGS=-I/home/user1/util/hdf5_1.10.8/include ¥
              LDFLAGS=-L/home/user1/util/hdf5_1.10.8/lib
$ make
$ make install
```

4.3 Installation of PPS Toolkit (TKIO)

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

4.3.1 Download

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

4.3.2 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
```

4.3.3 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
```

4.3.4 Editing the Preferences File

Create a file to define environment variables. An example is shown below. Define environment variables that suit your environment. Please specify “disable optimization” (-O0) in the Fortran compiler options (FFLAGS). Linking errors may occur when building the sample program.

```
ulimit -s unlimited
export TKDEBUG=”-g”

export TKIO=/home/user1/util/tkio-3.102_gcc_ifx/tkio_v8
export HDF5_INC=/home/user1/util/hdf5-1.10.8/include
export HDF5_LIB=/home/user1/util/hdf5-1.10.8/lib
export NETCDF_INC=/home/user1/util/netcdf-c-4.9.2/include
export NETCDF_LIB=/home/user1/util/netcdf-c-4.9.2/lib
export CLASSPATH=$TKIO/classes
export JPEG_INC=/usr/include
export JPEG_LIB=/usr/lib64
export ZLIB=/usr/lib64
export LIBXML2_INC=/usr/include/libxml2
export LIBXML2_LIB=/usr/lib64

export
LD_LIBRARY_PATH=/home/user1/util/jdk1.8.0_60/lib:${ZLIB}:/home/user1/util/libtool
-2.4/lib:/home/user1/util/flex-2.5.39/lib:${LD_LIBRARY_PATH}

export
PATH=./:/home/user1/util/jdk1.8.0_60/bin:/home/user1/util/byacc-20150711/bin:/hom
e/user1/util/libtool-2.4/bin:/home/user1/util/flex-2.5.39/bin::${PATH}

export CC=gcc
export CFLAGS=’-fPIC -mcmodel=medium’
export CXXFLAGS=’-fPIC -mcmodel=medium’
export FFLAGS=’-O0 -fPIC -mcmodel=medium’
export FC=ifx
export F77=ifx
export F90=ifx
export FORTC=ifx

export PERL5LIB=/usr/share/perl5/vendor_perl/CPAN
```

4.3.5 Reading the Preferences File

The following command reads the preferences file.

$ source Preferences file name

4.3.6 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 NetCDF 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.

```
PROGRAM SAMPLE

#include "TKHEADERS.h"
#include "TK_1BKu.h"
#include "tkiofortdeclare.h"


RECORD /TKINFO/ granuleHandle1BKu


RECORD /L1BKu_FS/ L1BKu

       PARAMETER( NANGLE=49, NRANGE=260 )

       INTEGER status, numOfScan
       INTEGER iscan, iangle, irange
       CHARACTER*255 jobname, file
REAL*8 lat(NANGLE), lon(NANGLE)
       INTEGER*2 noisePower(NANGLE),
echoPower(NRANGE,NANGLE)
       INTEGER*2 year, msec, dayOfYear
       BYTE month, dayOfMon, hour, min, sec
       REAL*8 secOfDay

       call getarg( 1, jobname )
       call getarg( 2, file )

! Open the file for reading.
       status = TKopen( file,'1BKu,TKREAD, 'NETCDF4', jobname, granuleHandle1BKu, 0 )
       IF ( status .NE. TK_SUCCESS ) THEN
         status = TKmessage( granuleHandle1BKu.jobname, TKERROR,'message to print' )
       ENDIF


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

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".

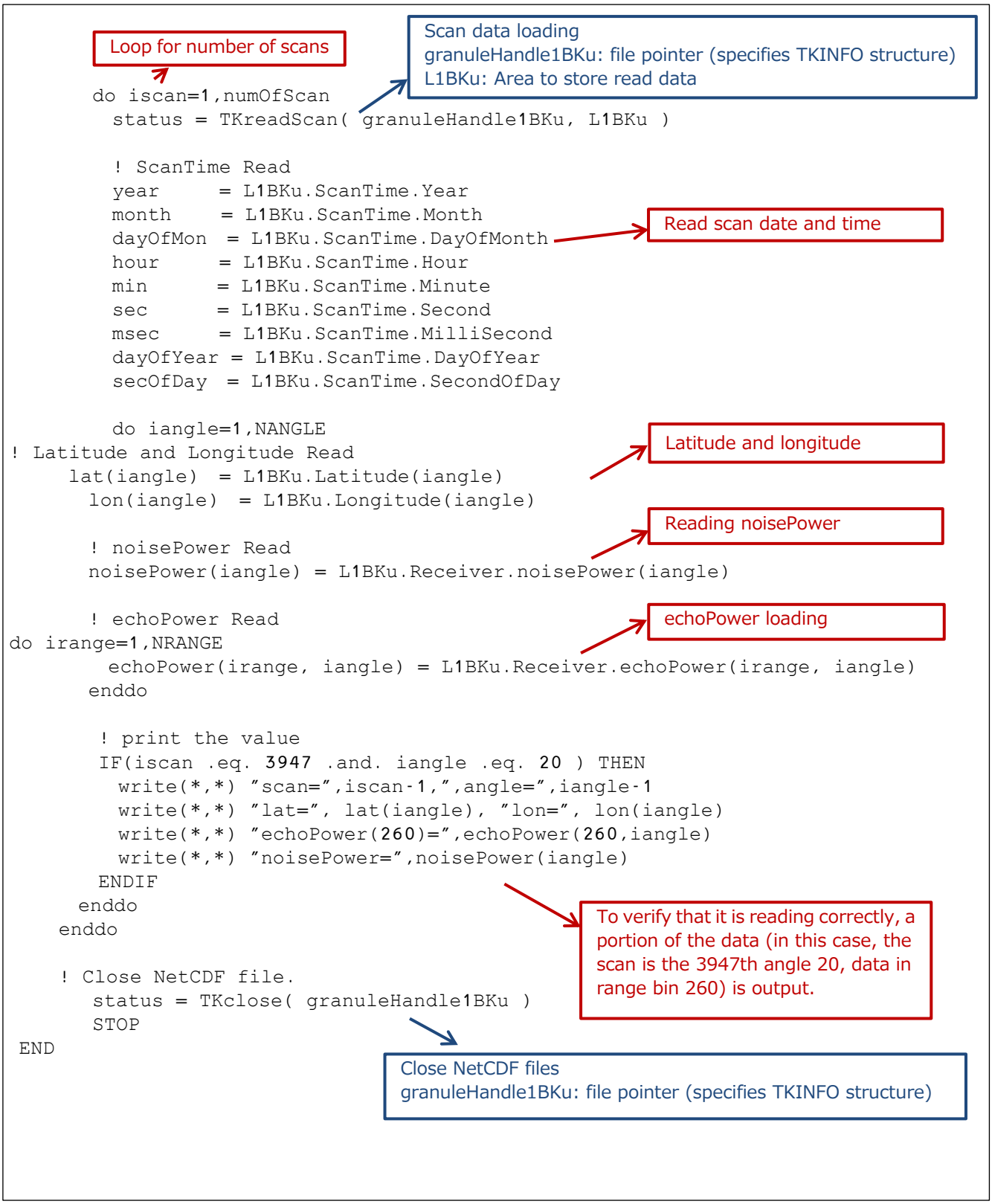
Always include in the case of Fortran.

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

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

Open NetCDF file
file:NetCDF file name (string specified in argument)
1BKu:Algorithm ID
TKREAD:Read designation
NETCDF4: 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 NetCDF 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




```
      do iscan=1,numOfScan
        status = TKreadScan( granuleHandle1BKu, L1BKu )

        ! ScanTime Read
        year      = L1BKu.ScanTime.Year
        month     = L1BKu.ScanTime.Month
        dayOfMon  = L1BKu.ScanTime.DayOfMonth
        hour      = L1BKu.ScanTime.Hour
        min       = L1BKu.ScanTime.Minute
        sec       = L1BKu.ScanTime.Second
        msec      = L1BKu.ScanTime.MilliSecond
        dayOfYear = L1BKu.ScanTime.DayOfYear
        secOfDay  = L1BKu.ScanTime.SecondOfDay

        do iangle=1,NANGLE
! Latitude and Longitude Read
     lat(iangle)  = L1BKu.Latitude(iangle)
      lon(iangle)  = L1BKu.Longitude(iangle)

      ! noisePower Read
      noisePower(iangle) = L1BKu.Receiver.noisePower(iangle)

      ! echoPower Read
do irange=1,NRANGE
        echoPower(irange, iangle) = L1BKu.Receiver.echoPower(irange, iangle)
      enddo

       ! print the value
       IF(iscan .eq. 3947 .and. iangle .eq. 20 ) THEN
         write(*,*) "scan=",iscan-1,",angle=",iangle-1
         write(*,*) "lat=", lat(iangle), "lon=", lon(iangle)
         write(*,*) "echoPower(260)=",echoPower(260,iangle)
         write(*,*) "noisePower=",noisePower(iangle)
       ENDIF
     enddo
    enddo

    ! Close NetCDF file.
      status = TKclose( granuleHandle1BKu )
      STOP
 END
```

5.1.2 Compilation Method

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

```
F90=ifx
MACHINE=LINUX
FFLAGS=-O0 -g -mcmodel=medium -fPIC -fpp -D$(MACHINE) -DLANGUAGE_FORTRAN
-DWRITEPIAINFO

MAIN=./sample_L1_Ku_F

INC = -I$(HDF5_INC) ¥
       -I$(NETCDF_INC) ¥
       -I$(TKIO)/inc/fortcode ¥
       -I$(LIBXML2_INC)

LIB = -L$(HDF5_LIB) ¥
       -L$(NETCDF_LIB) ¥
       -L$(TKIO)/lib

LIBES   = -ltkcselect ¥
          -ltkchdf5algs -ltkchdf5 ¥
          -ltkcnetcdf4 -ltkcnetcdf4algs -ltkfortalgmodules -ltkfortselect -ltkc ¥
          -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -lnetcdf

$(MAIN): $(MAIN).o
       $(F90) $(FFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).f90
       $(F90) $(FFLAGS) $(INC) -c $< -o $@

clean:
       rm -f *.o $(MAIN)
```

The name of the program in 5.1.1.

The path to the NetCDF include and library directories.

The path of the directory where the Fortran header files of the PPS Toolkit (TKIO) are located

The path of the directory where the PPS Toolkit (TKIO) libraries are located

5.1.3 Execution results

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

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

The second argument is the NetCDF file name.

```
$ ./sample_L1_Ku_F
TEST ../data/GPM/DPR/GPMCOR_KUR_2604010211_0345_068594_1BS_DUB_08A.nc
 scan=       3946 ,angle=         19
 lat=   64.7641448974609      lon=   57.4538764953613
 echoPower(260)= -29999
 noisePower= -11130
```

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 NetCDF 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.

```
PROGRAM SAMPLE

#include "TKHEADERS.h"
#include "TK_2ADPR.h"

#include "tkiofortdeclare.h"


RECORD /TKINFO/ granuleHandleL2DPR


RECORD /L2ADPR_SWATHS/ L2ADPR

       PARAMETER( NANGLE=49, NRANGE=176 )

       INTEGER status, numOfScan
       INTEGER iscan, iangle, irange
       CHARACTER*255 jobname, file
REAL*8 lat(NANGLE), lon(NANGLE)
REAL*4 precipRateESurface(NANGLE)
REAL*4 precipWater(NRANGE,NANGLE)
       INTEGER*2 year, msec, dayOfYear
       BYTE month, dayOfMon, hour, min, sec
       REAL*8 secOfDay

       call getarg( 1, jobname )
       call getarg( 2, file )

!Open the file for reading.
       status = TKopen( file,'2ADPR,TKREAD,'NETCDF4',jobname, granuleHandleL2DPR, 0 )
       IF ( status .NE. TK_SUCCESS ) THEN
         status = TKmessage( granuleHandleL2DPR.jobname, TKERROR,'message to print' )
       ENDIF

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

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 NetCDF files and is used when using the PPS Toolkit.

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

Open NetCDF file
file:NetCDF file name (string specified in argument)
2ADPR: Algorithm ID
TKREAD:Read designation
NETCDF4: Format type
jobname:job name (string specified by argument)
granuleHandleL2DPR: file pointer (specifies TKINFO structure)
1: Internal compression of files (specify 1)

Metadata read (scan count readout)
granuleHandle2ADPR: file pointer (specifies TKINFO structure)
FS_SwathHeader": Item name in the NetCDF 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.
numOfScan: Variable that stores the number of scans read

```
        do iscan=1,numOfScan
          status = TKreadScan( granuleHandleL2DPR, L2ADPR )

          ! ScanTime Read
          year      = L2ADPR.FS.ScanTime.Year
          month     = L2ADPR.FS.ScanTime.Month
          dayOfMon  = L2ADPR.FS.ScanTime.DayOfMonth
          hour      = L2ADPR.FS.ScanTime.Hour
          min       = L2ADPR.FS.ScanTime.Minute
          sec       = L2ADPR.FS.ScanTime.Second
          msec      = L2ADPR.FS.ScanTime.MilliSecond
          dayOfYear = L2ADPR.FS.ScanTime.DayOfYear
          secOfDay  = L2ADPR.FS.ScanTime.SecondOfDay

          do iangle=1,NANGLE

! Latitude and Longitude Read
        lat(iangle)  = L2ADPR.FS.Latitude(iangle)
        lon(iangle)  = L2ADPR.FS.Longitude(iangle)

        ! precipRateESurface Read
        precipRateESurface(iangle) = L2ADPR.FS.SLV.precipRateESurface(iangle)


do irange=1,NRANGE
          precipWater(irange, iangle) = L2ADPR.FS.SLV. precipWater (irange, iangle)
        enddo



        ! print the value
        IF(iscan .eq. 5211 .and. iangle .eq. 44 ) THEN
          write(*,*) "scan=",iscan-1,",angle=",iangle-1
          write(*,*) "lat=", lat(iangle), "lon=", lon(iangle)
          write(*,*) "precipRateESurface=",precipRateESurface (iangle), "(mm/hr)"
          write(*,*) "precipWater(171,44)=", precipWater(171, iangle), "(g/m~3)"
        ENDIF
      enddo
    enddo

        status = TKclose( granuleHandleL2DPR )
        STOP
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 5211nd angle 44 data) is output.

Close NetCDF files
granuleHandleL2DPR: file pointer (specifies TKINFO structure)

5.2.2 Compilation Method

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

```
F90=ifx
MACHINE=LINUX
FFLAGS=-O0 -g -mcmodel=medium -fPIC -fpp -D$(MACHINE) -DLANGUAGE_FORTRAN
-DWRITEPIAINFO

MAIN=./sample_L2_DPR_F

INC = -I$(HDF5_INC) ¥
       -I$(NETCDF_INC) ¥
       -I$(TKIO)/inc/fortcode ¥
       -I$(LIBXML2_INC)

LIB = -L$(HDF5_LIB) ¥
       -L$(NETCDF_LIB) ¥
       -L$(TKIO)/lib

LIBES   = -ltkcselect ¥
          -ltkchdf5algs -ltkchdf5 ¥
          -ltkcnetcdf4 -ltkcnetcdf4algs -ltkfortalgmodules -ltkfortselect -ltkc ¥
          -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -lnetcdf

$(MAIN): $(MAIN).o
       $(F90) $(FFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).f90
       $(F90) $(FFLAGS) $(INC) -c $< -o $@

clean:
       rm -f *.o $(MAIN)
```

The name of the program in 5.2.1.

The path to the NetCDF include and library directories.

The path of the directory where the Fortran header files of the PPS Toolkit (TKIO) are located

The path of the directory where the PPS Toolkit (TKIO) libraries are located

5.2.3 Execution results

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

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

The second argument is the NetCDF file name.

```
$ ./sample_L2_DPR_F
TEST ../data/GPM/DPR/GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc
 scan=       5210 ,angle=         43
 lat=   32.0528602600098      lon=   132.983184814453
 precipRateESurface=   2.261208     (mm/hr)
 precipWater(171,44)=  0.1282614     (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 NetCDF file as arguments and reads the data named precipRateESurface.mean from the file specified as the argument.

```
PROGRAM SAMPLE

#include "TKHEADERS.h"
#include "TK_3DPR.h"
#include "tkiofortdeclare.h"

RECORD /TKINFO/ granuleHandle3DPR
RECORD /L3DPR_FS/ L3DPR

       INTEGER status
       INTEGER st, rt, chn, lnL, ltL
       CHARACTER*255 jobname, file
   REAL*4 precipEsurf(28,72,3,3,3)
   REAL*4 lat, lon
       call getarg( 1, jobname )
       call getarg( 2, file )

   ! Open the file for reading.
status = TKopen( file, '3DPR', TKREAD, 'NETCDF4', jobname, granuleHandle3DPR, 0 )
IF ( status .NE. TK_SUCCESS ) THEN
  status = TKmessage( granuleHandle3DPR.jobname, TKERROR,'message to print' )
ENDIF

   ! Grid data read
   status = TKreadGrid( granuleHandle3DPR, L3DPR )

   do ltL=1, 28
     do lnL=1, 72
       do chn=1, 3
         do rt=1, 3
           do st=1, 3
             ! precipRateESurface.mean Read
             precipEsurf(ltL, lnL, chn, rt, st) =
L3DPR.G1.precipRateESurface.mean(ltL, lnL, chn, rt, st)

             ! print the value
             IF(lnL .eq. 64 .and. ltL .eq. 15 .and. chn .eq. 1 .and. st .eq. 1) THEN
               write(*,*) "st=",st-1,"rt=",rt-1
               lat = (140.0/28.0) * (ltL-1) - 70.0 + (140.0/28.0/2)
               lon = (360.0/72.0) * (lnL-1) - 180.0 + (360.0/72.0/2)
               write(*,*) "lat=",lat,"lon=",lon
               write(*,*) "chn=0 lnL=63 ltL=14
precipRateEsurface.mean=",precipEsurf(ltL, lnL, chn, rt, st), "(mm/hr)"
             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 NetCDF files and is used when using the

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

Open NetCDF file
file:NetCDF file name (string specified in argument)
3DPR: Algorithm ID, TKREAD: Read specification
NETCDF4: 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.

```
          enddo
        enddo
      enddo
    enddo
  enddo

  ! HDF file close
      status = TKclose( granuleHandle3DPR )
      STOP
END
```

Close NetCDF 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.

```
F90=ifx
MACHINE=LINUX
FFLAGS=-O0 -g -mcmodel=medium -fPIC -fpp -D$(MACHINE) -DLANGUAGE_FORTRAN
-DWRITEPIAINFO

MAIN=./sample_L3_DPR_F

INC = -I$(HDF5_INC) ¥
      -I$(NETCDF_INC) ¥
      -I$(TKIO)/inc/fortcode ¥
      -I$(LIBXML2_INC)

LIB = -L$(HDF5_LIB) ¥
      -L$(NETCDF_LIB) ¥
      -L$(TKIO)/lib

LIBES   = -ltkcselect ¥
          -ltkchdf5algs -ltkchdf5 ¥
          -ltkcnetcdf4 -ltkcnetcdf4algs -ltkfortalgmodules -ltkfortselect -ltkc ¥
          -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -lnetcdf

$(MAIN): $(MAIN).o
      $(F90) $(FFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).f90
      $(F90) $(FFLAGS) $(INC) -c $< -o $@

clean:
      rm -f *.o $(MAIN)
```

The name of the program in 5.3.1.

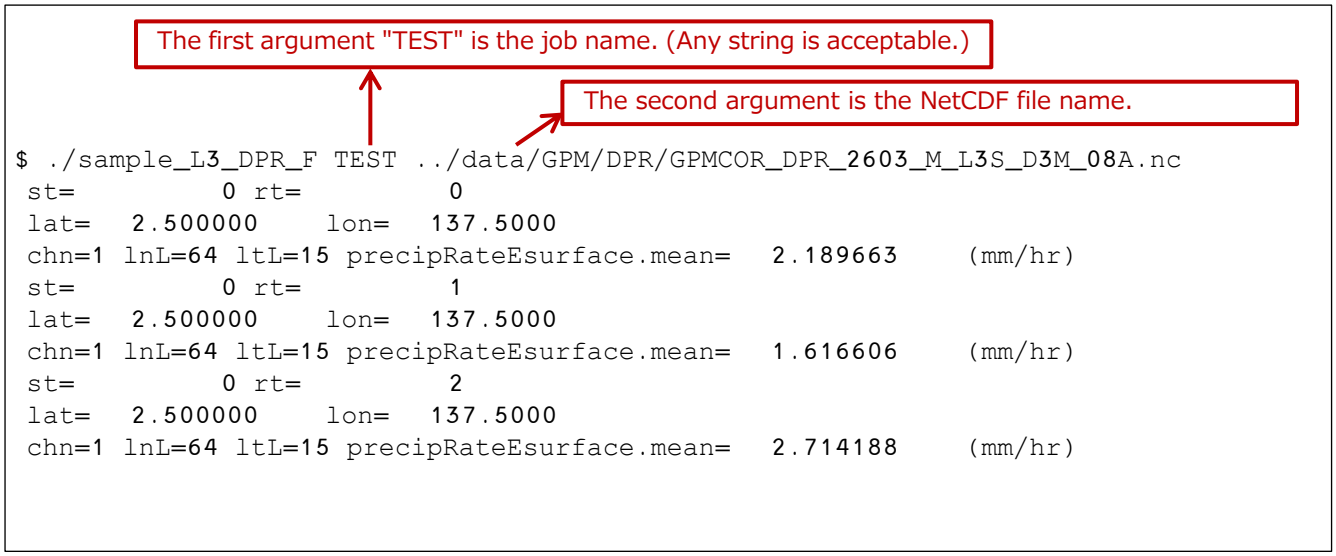
The path to the NetCDF include and library directories.

The path of the directory where the header files for Fortran of the PPS Toolkit (TKIO) are located

The path of the directory where the PPS Toolkit (TKIO) libraries are located

5.3.3 Execution results

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




```
$ ./sample_L3_DPR_F TEST ../data/GPM/DPR/GPMCOR_DPR_2603_M_L3S_D3M_08A.nc
 st=          0 rt=          0
 lat=   2.500000    lon=   137.5000
 chn=1 lnL=64 ltL=15 precipRateEsurface.mean=   2.189663    (mm/hr)
 st=          0 rt=          1
 lat=   2.500000    lon=   137.5000
 chn=1 lnL=64 ltL=15 precipRateEsurface.mean=   1.616606    (mm/hr)
 st=          0 rt=          2
 lat=   2.500000    lon=   137.5000
 chn=1 lnL=64 ltL=15 precipRateEsurface.mean=   2.714188    (mm/hr)
```

5.4 About the version of PPS Toolkit (TKIO)

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

To find out the version of an NetCDF file, use PPS Viewer THOR to read the FileInfo of the NetCDF 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".

It is this section that refers to the contents of TK_2ADPR.h.
Check the contents of "TK_2ADPR_bk2.h" for the corresponding
Change to "L2ADPR_SWATHS_bk2".

```
 22:    call getarg( 1, jobname )
 23:    call getarg( 2, file )
 24:
 25: !Open the file for reading.
 26:    status = TKopen(file,'2ADPR',TKREAD,'NETCDF4',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.

```
PROGRAM SAMPLE

      use hdf5   ! This module contains all necessary modules

      CHARACTER(LEN=34), PARAMETER :: filename = "../data/GPM/DPR/
GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc"

      CHARACTER(LEN=26), PARAMETER :: dsetname = "/HS/SLV/precipRateESurface"

      INTEGER(HID_T) :: file_id      ! File identifier
      INTEGER(HID_T) :: dset_id      ! Dataset identifier
      REAL*4, DIMENSION(49,7988) :: precipRateESurface ! Data buffers
      INTEGER(HSIZE_T), DIMENSION(2) :: data_dims
      INTEGER error

      ! Initialize FORTRAN interface.
      CALL h5open_f(error)

      ! Open an existing file.
      CALL h5fopen_f (filename, H5F_ACC_RDWR_F, file_id, error)
```

The name of the NetCDF file to be read.

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

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

NetCDF file open
filename: NetCDF 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)

```
        ! Open an existing dataset.
         CALL h5dopen_f(file_id, dsetname, dset_id, error)

        ! Read precipRateESurface.
```

data loading
dset_id: Specify the value output by h5dopen_f.

```
       data_dims(1) = 49
        data_dims(2) = 7988
        CALL h5dread_f(dset_id, H5T_NATIVE_REAL, precipRateESurface, data_dims,
error)

        ! print the value
         write(*,*) "filename=",filename,""
         write(*,*) "dataset name=",dsetname,""
        write(*,*) "precipRateESurface(44,5211)=",precipRateESurface(44,5211),
"(mm/hr)"

         ! Close the dataset.
         CALL h5dclose_f(dset_id, error)

         ! Close the file.
         CALL h5fclose_f(file_id, error)

         ! Close FORTRAN interface.
         CALL h5close_f(error)

       STOP
  END
```

6.1.2 Compilation Method

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

```
F90=gfortran
MACHINE=LINUX
FFLAGS=-g -mcmodel=medium -fPIC -cpp -fdec-structure -D$(MACHINE) -DLANGUAGE_FORTRAN
-DWRITEPIAINFO

MAIN=./sample_HDF5_L2_DPR_F

INC = -I$(HDF5_INC) ¥
      -I$(NETCDF_INC) ¥
      -I$(LIBXML2_INC)

LIB = -L$(HDF5_LIB) ¥
      -L$(NETCDF_LIB)

LIBES = -lnetcdf -lm -lhdf5_hl_fortran -lhdf5_fortran -lhdf5_hl -lhdf5 -ljpeg -lxml2

$(MAIN): $(MAIN).o
      $(F90) $(FFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).f90
      $(F90) $(FFLAGS) $(INC) -c $< -o $@

clean:
      rm -f *.o $(MAIN)
```

Name of the program in 6.1.1.

The path to the NetCDF 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=../data/GPM/DPR/GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc
 dataset name=/FS/SLV/precipRateESurface
 precipRateESurface(44,5211)=   2.261208     (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.

```
PROGRAM SAMPLE

    use hdf5   ! This module contains all necessary modules

    CHARACTER(LEN=38), PARAMETER :: filename =
"../data/GPM/DPR/GPMCOR_DPR_2603_M_L3S_D3M_08A.nc"

    CHARACTER(LEN=34), PARAMETER :: dsetname = "/FS/G1/precipRateESurface/mean"

    INTEGER(HID_T) :: file_id       ! File identifier
    INTEGER(HID_T) :: dset_id       ! Dataset identifier
    REAL*4, DIMENSION(28,72,3,3,3) :: precipRateESurface ! Data buffers
    REAL*4 lat, lon
    INTEGER(HSIZE_T), DIMENSION(5) :: data_dims
       INTEGER error

    ! Initialize FORTRAN interface.
     CALL h5open_f(error)

    ! Open an existing file.
    CALL h5fopen_f (filename, H5F_ACC_RDWR_F, file_id, error)

    ! Open an existing dataset.
    CALL h5dopen_f(file_id, dsetname, dset_id, error)
```

The name of the NetCDF file to be read.

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

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

NetCDF file open
filename: NetCDF 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)

```
      ! Read precipRateESurface.
      data_dims(1) = 28
      data_dims(2) = 72
      data_dims(3) = 3
      data_dims(4) = 3
      data_dims(5) = 3
      CALL h5dread_f(dset_id, H5T_NATIVE_REAL, precipRateESurface, data_dims, error)

      ! print the value
      write(*,*) "filename=",filename
      write(*,*) "dataset name=",dsetname
      lat = (140.0/28.0) * (15-1) - 70.0 + (140.0/28.0/2)
      lon = (360.0/72.0) * (64-1) - 180.0 + (360.0/72.0/2)
      write(*,*) "lat=",lat,"lon=",lon
      write(*,*)
"precipRateESurface.mean(15,64,1,1,1)=",precipRateESurface(15,64,1,1,1)
      write(*,*)
"precipRateESurface.mean(15,64,1,2,1)=",precipRateESurface(15,64,1,2,1)

      ! Close the dataset.
      CALL h5dclose_f(dset_id, error)

      ! Close the file.
      CALL h5fclose_f(file_id, error)

      ! Close FORTRAN interface.
      CALL h5close_f(error)

       STOP
  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.

```
F90=gfortran
MACHINE=LINUX
FFLAGS=-g -mcmodel=medium -fPIC -cpp -fdec-structure -D$(MACHINE) -DLANGUAGE_FORTRAN
-DWRITEPIAINFO

MAIN=./sample_HDF5_L3_DPR_F

INC = -I$(HDF5_INC) ¥
      -I$(NETCDF_INC) ¥
      -I$(LIBXML2_INC)

LIB = -L$(HDF5_LIB) ¥
      -L$(NETCDF_LIB)

LIBES = -lnetcdf -lm -lhdf5_hl_fortran -lhdf5_fortran -lhdf5_hl -lhdf5 -ljpeg -lxml2

$(MAIN): $(MAIN).o
      $(F90) $(FFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).f90
      $(F90) $(FFLAGS) $(INC) -c $< -o $@

clean:
      rm -f *.o $(MAIN)
```

The name of the program in 6.2.1.

The path to the NetCDF 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=../data/GPM/DPR/GPMCOR_DPR_2603_M_L3S_D3M_08A.nc
 dataset name=/FS/G1/precipRateESurface/mean
 lat=   2.50000000     lon=   137.500000
 precipRateESurface.mean(15,64,1,1,1)=   2.18966341
 precipRateESurface.mean(15,64,1,2,1)=   1.61660624
```

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 NetCDF file and how to create a Fortran program to read that binary file.

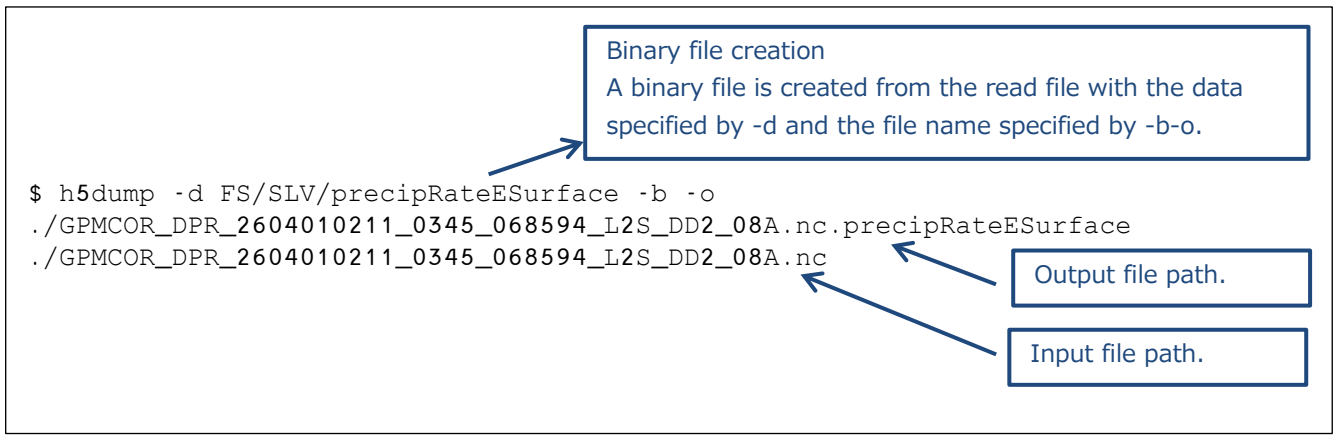
Explanations in red describe sample programs.

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 binary file using h5dump.



When the above command is executed, the following is displayed and the binary file is created in the output file path.

```
HDF5 "./GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc" {
DATASET "FS/SLV/precipRateESurface" {
   DATATYPE  H5T_IEEE_F32LE
   DATASPACE  SIMPLE { ( 7988, 49 ) / ( H5S_UNLIMITED, 49 ) }
   DATA {
   }
}
}
```

7.1.2 Source Programs

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

```
program sample

    CHARACTER(LEN=66), PARAMETER :: filename =”../data/GPM/DPR/
GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc.precipRateESurface ”

    ! read data area
    real*4 precipRateESurface(49,7988)



    ! binary file open
    open(10,file=filename,access=’direct’,status=’old’,recl=4*49*7988)


    ! binary file read
    read(10,rec=1) precipRateESurface

    ! print the value
    write(*,*) ”filename=”,filename,””
    write(*,*) ”precipRateESurface(44,5211)=”,precipRateESurface(44,5211),
”(mm/hr)”
    write(*,*) ”precipRateESurface(45,5211)=”,precipRateESurface(45,5211),
”(mm/hr)”

    ! binary file close
    close(10)

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.

```
F90=gfortran
MACHINE=LINUX
FFLAGS=-g -mcmodel=medium -fPIC -cpp -fdec-structure -D$(MACHINE) -DLANGUAGE_FORTRAN
-DWRITEPIAINFO

MAIN=./sample_h5dump_L2_F

INC = -I$(HDF5_INC) ¥
      -I$(NETCDF_INC) ¥
      -I$(LIBXML2_INC)

LIB = -L$(HDF5_LIB) ¥
      -L$(NETCDF_LIB)

LIBES = -lnetcdf -lm -lhdf5_hl_fortran -lhdf5_fortran -lhdf5_hl -lhdf5 -ljpeg -lxml2
-ltirpc

$(MAIN): $(MAIN).o
      $(F90) $(FFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).f90
      $(F90) $(FFLAGS) $(INC) -c $< -o $@

clean:
      rm -f *.o $(MAIN)
```

The name of the program in 7.1.1.

The path to the NetCDF include and library directories.

7.1.4 Execution Results

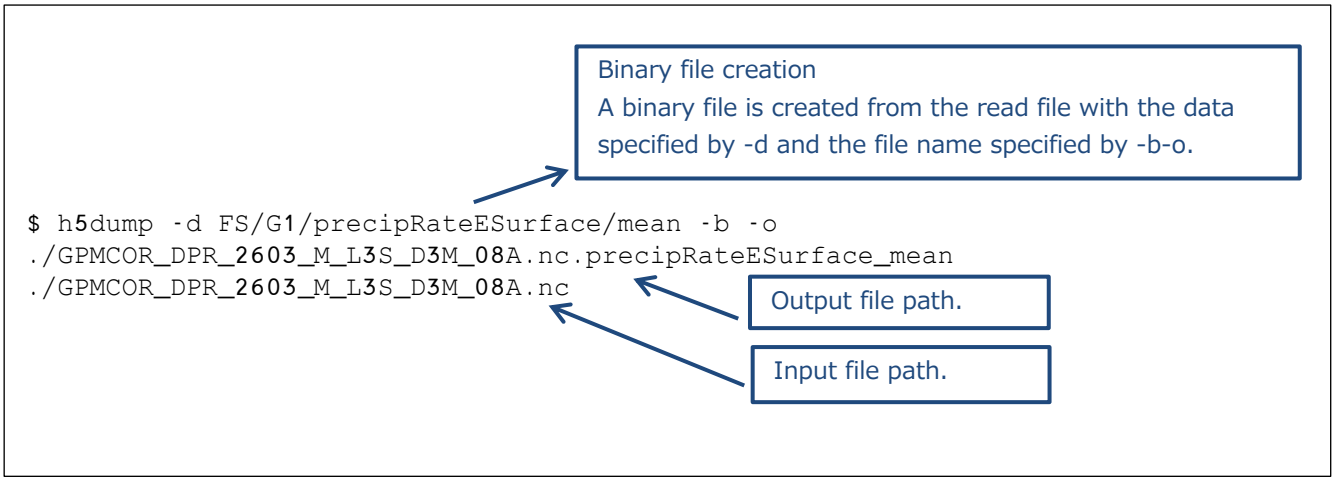
The following are the results of executing the program described in 7.1.2.

```
$ ./sample_h5dump_L2_F

filename=../data/GPM/DPR/GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc.precipR
ateESurface

 precipRateESurface(44,5211)=   2.26120758     (mm/hr)
 precipRateESurface(45,5211)=   2.95193911     (mm/hr)
```

7.2 L3 data reading

7.2.1 Creating Binary Files

Here is an example of creating a binary file using h5dump.



When the above command is executed, the following is displayed and the binary file is created in the output file path.

```
HDF5 "./GPMCOR_DPR_2603_M_L3S_D3M_08A.nc" {
DATASET "FS/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.

```
program sample

    CHARACTER(LEN=74), PARAMETER :: filename =" ../data/GPM/DPR&
&/GPMCOR_DPR_2603_M_L3S_D3M_08A.nc.precipRateESurface_mean"

    ! read data area
    real*4 precipRateESurface(28,72,3,3,3)

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

    ! binary file read
    read(10,rec=1) precipRateESurface

    ! print the value
    write(*,*) "filename=",filename
    lat = (140.0/28.0) * (15-1) - 70.0 + (140.0/28.0/2)
    lon = (360.0/72.0) * (64-1) - 180.0 + (360.0/72.0/2)
    write(*,*) "lat=",lat,"lon=",lon
    write(*,*) "precipRateESurface.mean(15,64,1,1,1)=",
precipRateESurface(15,64,1,1,1)
   write(*,*) "precipRateESurface.mean(15,64,1,2,1)=",
precipRateESurface(15,64,1,2,1)

    ! binary file close
    close(10)

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.

```
F90=gfortran
MACHINE=LINUX
FFLAGS=-g -mcmodel=medium -fPIC -cpp -fdec-structure -D$(MACHINE) -DLANGUAGE_FORTRAN
-DWRITEPIAINFO

MAIN=./sample_h5dump_L3_F

INC = -I$(HDF5_INC) ¥
      -I$(NETCDF_INC) ¥
      -I$(LIBXML2_INC)

LIB = -L$(HDF5_LIB) ¥
      -L$(NETCDF_LIB)

LIBES = -lnetcdf -lm -lhdf5_hl_fortran -lhdf5_fortran -lhdf5_hl -lhdf5 -ljpeg -lxml2
-ltirpc

$(MAIN): $(MAIN).o
      $(F90) $(FFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).f90
      $(F90) $(FFLAGS) $(INC) -c $< -o $@

clean:
      rm -f *.o $(MAIN)
```

The name of the program in 7.2.1.

The path to the NetCDF include and library directories.

7.2.4 Execution Results

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

```
./sample_h5dump_L3_F
 filename=../data/GPM/DPR/GPMCOR_DPR_2603_M_L3S_D3M_08A.nc.precipRateESurface_mean

 lat=   2.50000000     lon=   137.500000
 precipRateESurface.mean(15,64,1,1,1)=   2.18966341
 precipRateESurface.mean(15,64,1,2,1)=   1.61660624
```

8. Revision history

revision history

version number	Date	Revised contents	remarks
1	Jan 26, 2016		
2	Sep 13, 2017	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	Mar 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	Dec 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	Dec 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	Jan 21, 2022	5.-7. Corrections due to modification of sample programs (review of display position, addition of lat/lon to display items, etc.)	
8	Feb 4, 2022	Error Correction	
9	Jun 1, 2026	Sample data updated to V8 Corrected code description to match V8 and NetCDF	