# GPM/TRMM Data reading program guide (C language version)

2026/06/01

7th ed.

This document describes how to create a program (in C language) to read data from the Global Precipitation Measurement Satellite (GPM/TRMM).

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 explains how to read in GPM/TRMM data using the C 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 the C language, there are other methods to read GPM/TRMM data as shown in Table 1.1.
To determine which method to use, please refer to the "Read Method Judgment Flow" on the next page.
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 Loading 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) | |

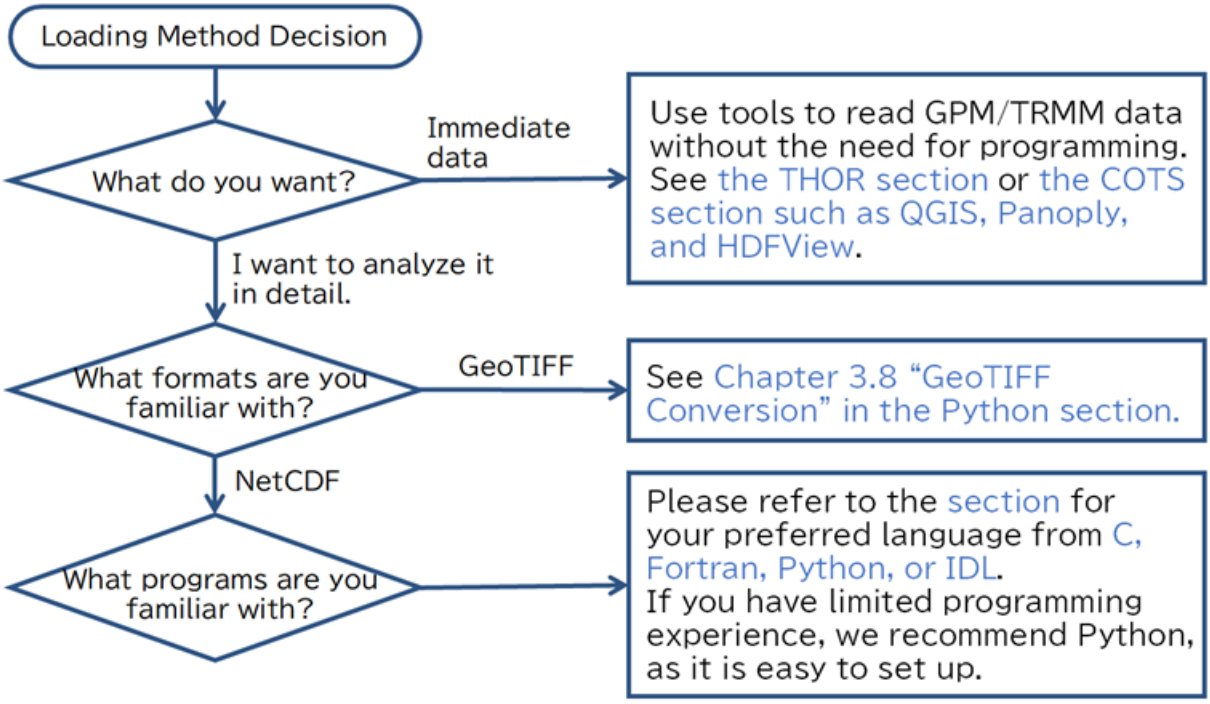


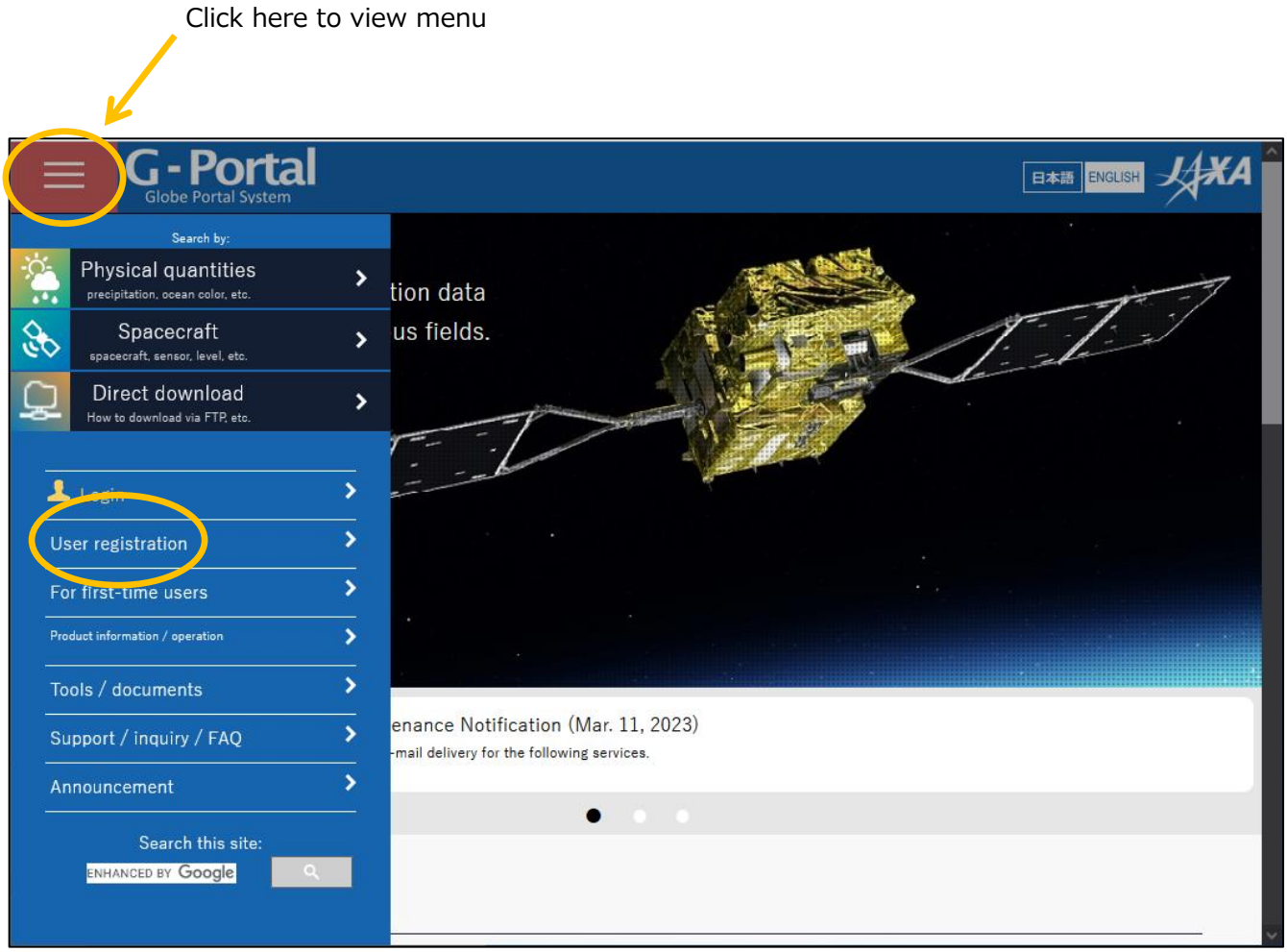

**Table 1.2 Sample Program Operation Check Table**

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

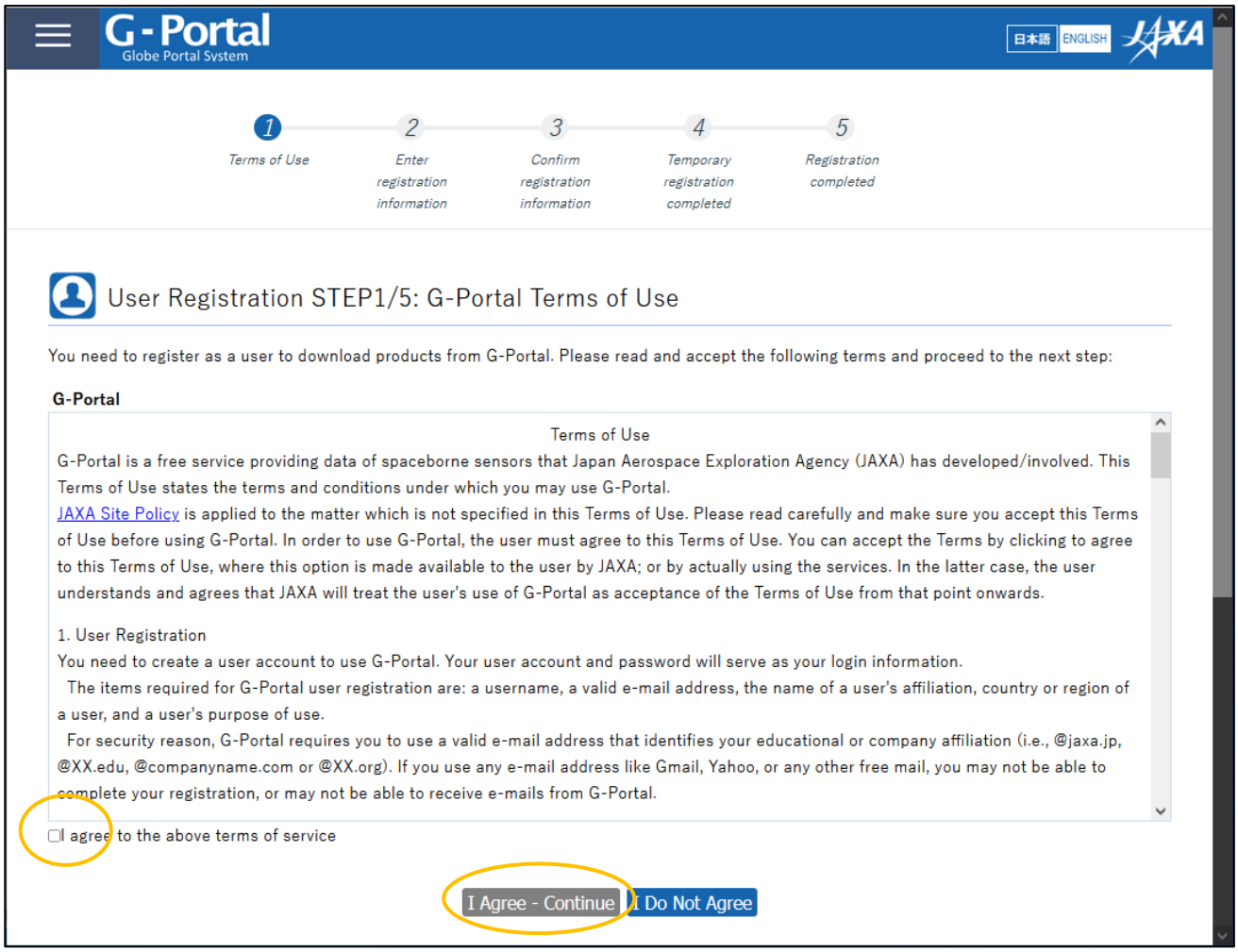
○ : Operation is confirmed. －: Operation is unconfirmed.

## 2. how to obtain GPM/TRMM data

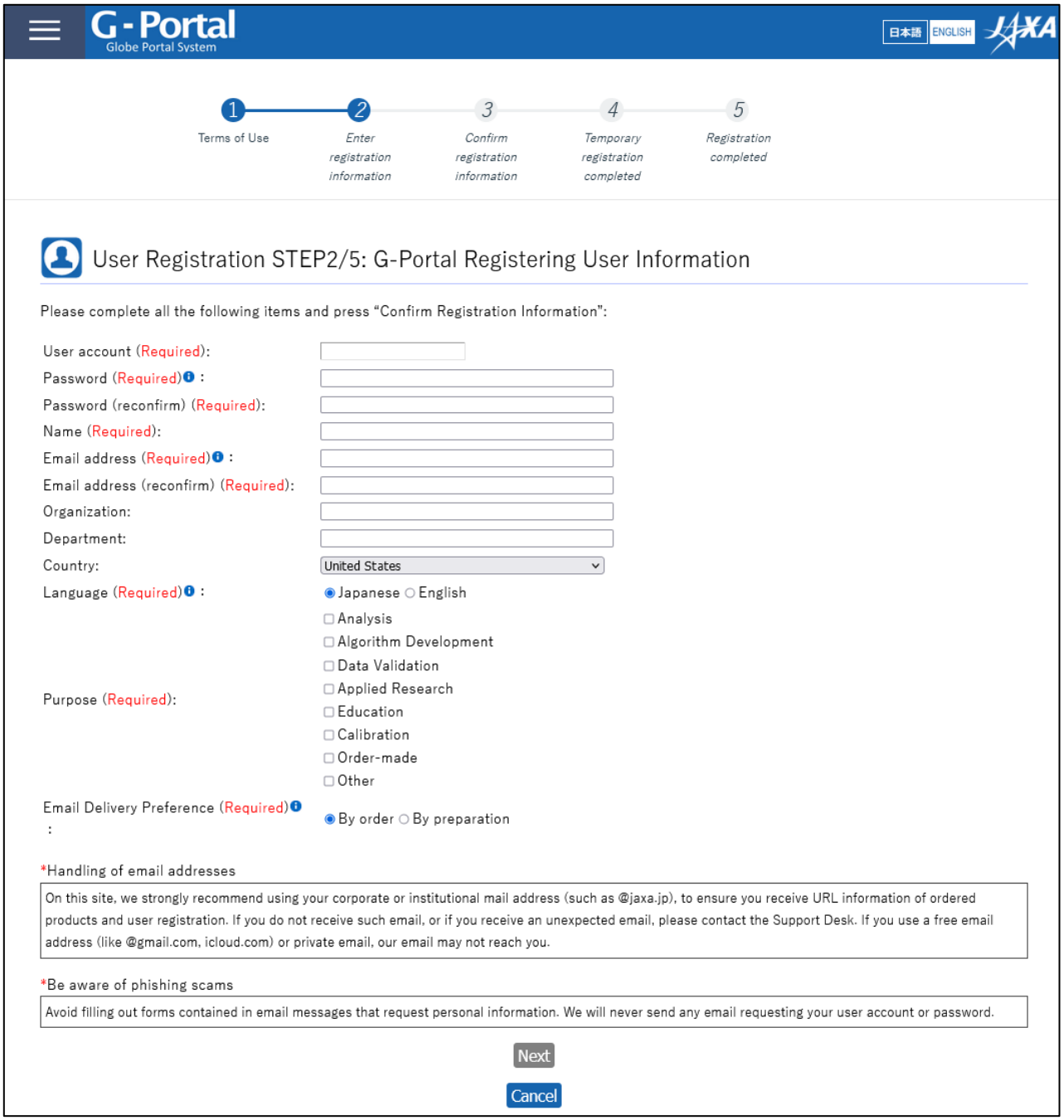
GPM/TRMM data can be obtained from the G-Portal site (https://www.gportal.jaxa.jp/gp/top.html). User registration is required to obtain the data, so please register by selecting "User Registration" from the menu on the G-Portal site.



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.

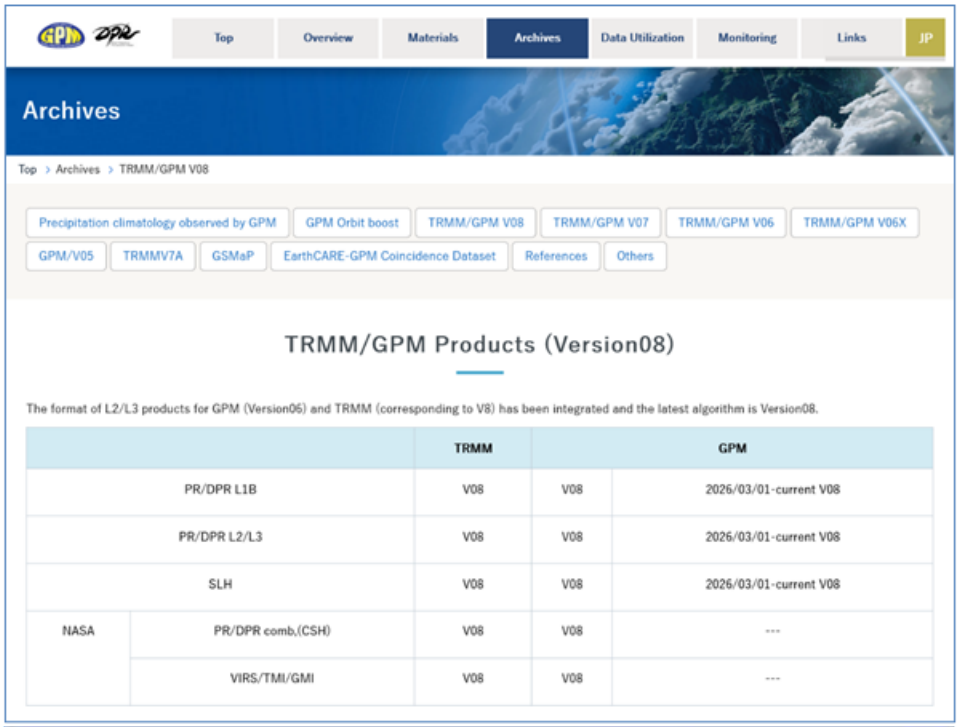
# 3. how to obtain related documents and sample programs

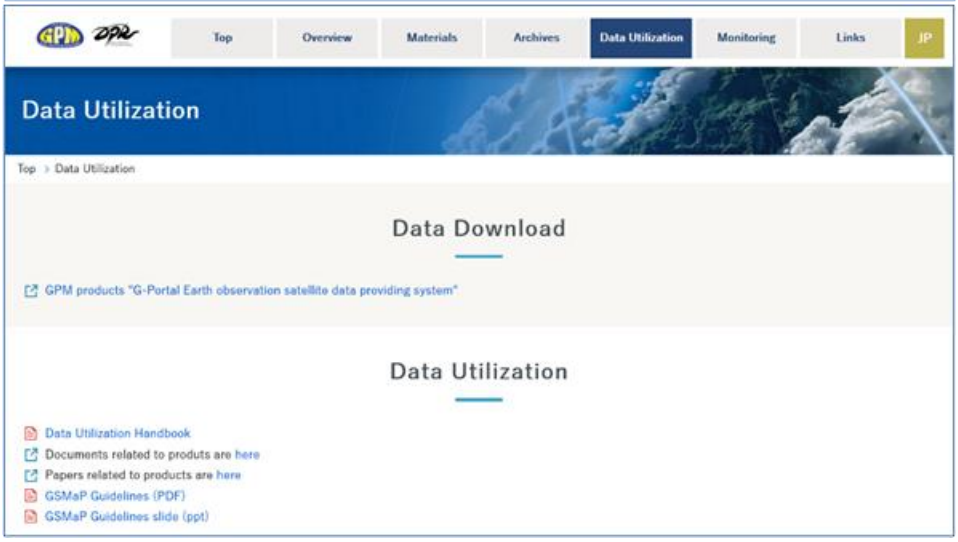
There are two types of documents related to GPM/TRMM data: documents related to data use and documents related to products. Both documents can be downloaded from the Global Precipitation Measurement Project (GPM) website ( https://www.eorc.jaxa.jp/GPM/en/index.html ). You can also download the sample 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_C.c | GPMCOR_KUR_2604010211_0345_068594_1BS_DUB_08A.nc |
| L2DPR | sample_L2_DPR_C.c | GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc |
| | sample_HDF5_L2_DPR_C.c | |
| | sample_h5dump_L2_C.c | |
| L3DPR | sample_L3_DPR_C.c | GPMCOR_DPR_2603_M_L3S_D3M_08A.nc |
| | sample_HDF5_L3_DPR_C.c | |
| | sample_h5dump_L3_C.c | |
| L2GMI | sample_L2_GMI_C.c | GPMCOR_GMI_2604010211_0345_068594_L2S_GL2_08A.nc |
| L3GMI | sample_L3_GMI_C.c | GPMCOR_GMI_2603_M_L3S_GL3_08A.nc |
| L2SLP | sample_L2_SLP_C.c | GPMCOR_DPR_2604010211_0345_068594_L2S_SLP_08A.nc |
| L3SLM | sample_L3_SLM_C.c | GPMCOR_DPR_2603_M_L3S_SLM_08A.nc |
| L3SLG | sample_L3_SLG_C.c | GPMCOR_DPR_2604010211_0345_068594_L3S_SLG_08A.nc |

# 4. installation of library tools

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

**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 | |

The confirmed product versions and PPS toolkit versions are as follows

**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

**Table 4.3 Operating Environment**

| (data) item | environment |
|---|---|
| calculator | Intel(R) Xeon(R) CPU E5-2665 0 @ 2.40GHz |
| OS | AlmaLinux release 8.8 |
| C compiler | gcc (GCC) 8.5.0 20210514 (Red Hat 8.5.0-28)<br>icx (ICX) Intel(R) oneAPI DPC++/C++ Compiler 2025.3.2 (2025.3.2.20260112) |
| HDF5 | hdf5-1.10.5 |
| NetCDF | netcdf-4.7.0 |
| PPS TKIO | tkio-3.102 |

## 4.1 Installation of HDF5

Execute the following commands to install the software.

$ dnf install hdf5 hdf5-devel

*Please ensure that the repository providing the HDF5 libraries is enabled beforehand. The sample program uses the libraries provided in the EPEL repository.

## 4.2 Installation of NetCDF

Execute the following commands to install the software.

$ dnf install netcdf netcdf-devel

*Please ensure that the repository containing the NetCDF libraries is enabled beforehand. The sample program uses the libraries provided in the EPEL repository.

## 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.

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

export TKIO=/home/user1/util/tkio-3.102/tkio_v8
export HDF5_INC=/usr/include
export HDF5_LIB=/usr/lib64
export NETCDF_INC=/usr/include
export NETCDF_LIB=/usr/lib64
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=’ -fPIC -mcmodel=medium’
export FC=gfortran
export F77=gfortran
export F90=gfortran
export FORTC=gfortran

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

Execute the following commands to compile.

$ . /INSTALL.pl compileJAVA

$ . /INSTALL.pl buildRW

$ . /INSTALL.pl compileRW

# 5. GPM/TRMM data reading with PPS Toolkit (TKIO)

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

Also, when creating a program using the PPS Toolkit (TKIO), you need to know the algorithm ID beforehand. The algorithm ID is an ID for each product (type of data) and is stored in the file header of the NetCDF file. Algorithm IDs of major products and examples of TKIO header files used are shown below. You can also check the file header information 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 | 3GSMAPH5 | TK_3GSMAPH5.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.

```
#include "TKheaders.h"
#include "TK_1BKu.h"

#define NRAY 49   /* number of Anglebin in one scan */
#define NBIN 260  /* number of Rangebin in one Anglebin */

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

    /* declare Data struct */
    TKINFO granuleHandle1BKu;
    L1BKu_FS  L1BKu;

    /* declare character */
    char job[256];
    char inputfile[256];

    strcpy(job, argv[1]);
    strcpy(inputfile, argv[2]);

    /* declare variables */
    int year, month, dayOfMon, hour, min;
    int i, j, k;
    int numOfScan;
    int status;

    /* declare array */
    float lat[NRAY], lon[NRAY];
    short noisePower[NRAY], echoPower[NRAY][NBIN];

    /* NetCDF file open */
    status = TKopen(inputfile, "1BKu", TKREAD, "NETCDF4", job, &granuleHandle1BKu, 1);
    if(status != TK_SUCCESS) {
        fprintf(stderr, "error:file open error[%s] input1BKu¥n", inputfile);
    }
```

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

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

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

job is used for the argument specified in TKopen.

The area to store data for one scan read is defined. To store the entire file, * the number of scans is required.

Open NetCDF file
inputfile:NetCDF file name (string specified by argument)
1BKu:Algorithm ID
TKREAD:Read designation
NETCDF4: Format type
job:job name (string specified by argument)
&granuleHandle1BKu: File pointer (specifies TKINFO structure)
1: Internal compression of files

Metadata read (scan count readout)
&granuleHandle1BKu: File pointer (specifies TKINFO structure)
"SwathHeader": The name of the item in the NetCDF file, which is 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: Address to store the number of scans read

```
    /* META data read */
    status = TKgetMetaInt(&granuleHandle1BKu, "SwathHeader",
                          "NumberScansGranule", &numOfScan);
    if(status != TK_SUCCESS) {
         fprintf(stderr, "error:file read error[%s]¥n", inputfile);
    }
```

Loop for number of scans

Scan data loading
&granuleHandle1BKu: File pointer (specifies TKINFO structure)
&L1BKu: Address to store the read data

```
    /* Scan data read */
    for (i=0; i<numOfScan; i++){
        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;

        for (j=0; j<NRAY; j++){
               /* Latitude and Longitude Read */
            lat[j]  = L1BKu.Latitude[j];
            lon[j]  = L1BKu.Longitude[j];

                /* noisePower Read */
             noisePower[j] = L1BKu.Receiver.noisePower[j];

                /* echoPower Read */
             for (k=0; k<NBIN; k++){
                echoPower[j][k] = L1BKu.Receiver.echoPower[j][k];
                }

                /* Print the value */
                if(i == 3946 && j == 19){
                   printf("Scan=%d, Angle=%d¥n", i, j);
                   printf("lat=%f, lon=%f¥n", lat[j], lon[j]);
                   printf("echoPower[259]= %d ¥n", echoPower[j][259]);
                   printf("noisePower= %d ¥n", noisePower[j]);
                }
        }
    }

    /* NetCDF File close*/
    status = TKclose(&granuleHandle1BKu);

    return 0;
}
```

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 NetCDF 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.

```
CC=icx
CFLAGS=-O0 -g -mcmodel=medium -fPIC

#CC=gcc
#CFLAGS=-g -mcmodel=medium -fpic -Wall

MAIN=./sample_L1_Ku_C

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

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

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

$(MAIN): $(MAIN).o
      $(CC) $(CFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).c
      $(CC) $(CFLAGS) $(INC) -c $< -o $@

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

If the Intel compiler is used as the compiler, lines 1 and 2 should be used and lines 3 and 4 should be comments.

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 header files for the C language 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.

```
$ ./sample_L1_Ku_C
TEST ../data/GPM/DPR/GPMCOR_KUR_2604010211_0345_068594_1BS_DUB_08A.nc

Scan=3946, Angle=19
lat=64.764145, lon=57.453876
echoPower[259]= -29999
noisePower= -11130
```

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

The second argument is the NetCDF file name.

## 5.2 L2 data reading

### 5.2.1 Source Programs

The following is an example program to read 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 zFactorCorrected data from the file specified as the argument.

```
#include "TKheaders.h"
#include "TK_2ADPR.h"

#define NRAY 49   /* number of Anglebin in one scan */
#define NBIN 176  /* number of Rangebin in one Anglebin */

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

   /* declare Data struct */
   TKINFO granuleHandle2ADPR;
   L2ADPR_SWATHS  L2ADPR;

   /* declare character */
   char job[256];
   char inputfile[256];

   strcpy(job, argv[1]);
   strcpy(inputfile, argv[2]);


   /* declare variables */
   int year, month, date, dayOfMon, hour, min;
   int i, j, k;
   int numOfScan;
   int status;



   /* declare array */
   float lat[NRAY], lon[NRAY], precipESurf[NRAY];
   float precipWater[NRAY][NBIN];

   /* NetCDF file open */
   status = TKopen(inputfile, "2ADPR", TKREAD, "NETCDF4", job, &granuleHandle2ADPR, 1);
   if(status != TK_SUCCESS) {
      fprintf(stderr, "error:file open error[%s] input2ADPR¥n", inputfile);
   }
```

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".

The granuleHandle2ADPR 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.

job is used for the argument specified in TKopen.

The area to store data for one scan read is defined. To store the entire file, * the number of scans is required.

Open NetCDF file
inputfile:NetCDF file name (string specified by argument)
2ADPR: Algorithm ID
TKREAD:Read designation
NETCDF4: Format type
job:job name (string specified by argument)
&granuleHandle2ADPR: 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": 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: Address to store the number of scans read

```
    /* META data read */
    status = TKgetMetaInt(&granuleHandle2ADPR, "FS_SwathHeader",
                          "NumberScansGranule", &numOfScan);
    if(status != TK_SUCCESS) {
        fprintf(stderr, "error:file read error[%s]¥n", inputfile);
    }
```

Loop for number of scans

Scan data loading
&granuleHandle2ADPR: File pointer (specifies TKINFO structure)
&L2ADPR: Address to store read data

```
   /* Scan data read */
    for (i=0; i<numOfScan; i++){
        status = TKreadScan( &granuleHandle2ADPR, &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;

        for (j=0; j<NRAY; j++){
                /* Latitude and Longitude Read */
            lat[j]  = L2ADPR.FS.Latitude[j];
            lon[j]  = L2ADPR.FS.Longitude[j];

                /* precipRateESurface Read */
            precipESurf[j] = L2ADPR.FS.SLV.precipRateESurface[j];
                for (k=0; k<NBIN; k++){
                    presipWater[j][k] = L2ADPR.FS.SLV.precipWater[j][k];
                }

                /* Print the value */
                if(i == 5210 && j == 43){
                    printf("Scan=%d, Angle=%d¥n", i, j);
                    printf("lat=%f, lon=%f¥n", lat[j], lon[j]);
                    printf("precipRateESurface= %f (mm/hr)¥n", precipESurf[j]);
                printf("precipWater[43][170]= %f (g/m~3)¥n", precipWater[j][170]);
                }
        }
    }

    /* NetCDF File close*/
    status = TKclose(&granuleHandle2ADPR);
    return 0;
}
```

Read scan date and time

Latitude and longitude

precipRateESurface loading

precipWater loading

To verify that it is being read correctly, a portion of the data (in this case, the data for angle 43, the 5210st scan) is output.

Close HDF files
&granuleHandle2ADPR: File pointer (specifies TKINFO structure)

### 5.2.2 Compilation Method

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

```
CC=icx
CFLAGS=-O0 -g -mcmodel=medium -fPIC

#CC=gcc
#CFLAGS=-g -mcmodel=medium -fpic -Wall

MAIN=./sample_L2_DPR_C

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

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

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

$(MAIN): $(MAIN).o
      $(CC) $(CFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).c
      $(CC) $(CFLAGS) $(INC) -c $< -o $@

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

If you use the GNU compiler, use lines 3 and 4 and leave lines 1 and 2 as comments.

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 header files for the C language 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.

```
$ ./sample_L2_DPR_C
TEST ../data/GPM/DPR/GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc
Scan=5210, Angle=43
lat=32.052860, lon=132.983185
precipRateESurface= 2.261208 (mm/hr)
precipWater[43][170]= 0.128261 (g/m~3)
```

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

The second argument is the NetCDF file name.

## 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.

```
#include "TKheaders.h"
#include "TK_3DPR.h"

#define LTL 28  /* number of low resolution 5° grid intervals of latitude from 70°S to 70°N. */
#define LNL 72  /* number of low resolution 5° grid intervals of longitude from 180°W to 180°E.
*/
#define CHN  3  /* number of channels: KuFS, KaMS, KaHS, DPRMS, KuMS. KaFS, DPRFS */
#define RT   3  /* number of rain types: all, stratiform, convective */
#define ST   3  /* number of surface types: all, ocean, land */

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

   /* declare Data struct */
   TKINFO granuleHandle3DPR;
   L3DPR_GRIDS  L3DPR;

   /* declare character */
   char job[256];
   char inputfile[256];

   strcpy(job, argv[1]);
   strcpy(inputfile, argv[2]);

   /* declare variables */
   int i, j, k, l, m, n;
   int status;
   float  lat,lon;
   /* declare array */
   float  precipESurf[ST][RT][CHN][LHL][LTL];

   /* NetCDF file open */
   status = TKopen(inputfile, "3DPR", TKREAD, "NETCDF4", job, &granuleHandle3DPR, 1);
   if(status != TK_SUCCESS) {
      fprintf(stderr, "error:file open error[%s] input3DPR¥n", inputfile);
   }
```

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".

granuleHandle3DPR is a structure that stores information on NetCDF files and is used when using the PPS Toolkit.

L3DPR is the structure of an NetCDF file, which contains data for one scan.

job is used for the argument specified in TKopen.

This defines the area where the read data is

Open NetCDF file
inputfile:NetCDF file name (string specified by argument)
3DPR: Algorithm ID
TKREAD:Read designation
NETCDF4: Format type
job:job name (string specified by argument)
&granuleHandle2BCMB: File pointer (specifies TKINFO structure)
1: Internal compression of files (specify 1)

```
   /* Grid data read */
   status = TKreadGridMulti( &granuleHandle3DPR, &L3DPR, "L3DPR_FS");

  /* precipRateESurface.mean data read */
   for (j=0; j<ST; j++){
          for (k=0; k<RT; k++){
          for (l=0; l<CHN; l++){
             for (m=0; m<LNL; m++){
                for (n=0; n<LTL; n++){
                       precipESurf[j][k][l][m][n] =
L3DPR.G1.precipRateESurface.mean[j][k][l][m][n];
                       if( m==63 && n==14 && l==0 && j==0 ) {
                       lat = (140.0/28.0) * n - 70.0 + (140.0/28.0/2);
                       lon = (360.0/72.0) * m - 180.0 + (360.0/72.0/2);
                       printf("lat=%f, lon=%f¥n",lat,lon);
                           printf("precipRateESurface.mean[%d][%d][%d][%d][%d}=%f (mm/hr)¥n",
                           j, k, l, m, n, precipESurf[j][k][l][m][n]);
                       }
                }
                }
             }
          }
   }

   /* NetCDF File close*/
   status = TKclose(&granuleHandle3DPR);

   return 0;
}
```

Grid data loading
&granuleHandle3DPR: File pointer (specifies TKINFO structure)
&L3DPR: Address to store read data
"L3DPR_FS": Data to be read

precipRateESurface.mean loading

To check that it is read correctly, a portion of the data (in this case, the 64th longitude grid interval, the 15th latitude grid interval, and the 1st channel) is output.

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.

```
CC=icx
CFLAGS=-O0 -g -mcmodel=medium -fPIC

#CC=gcc
#CFLAGS=-g -mcmodel=medium -fpic -Wall

MAIN=./sample_L3_DPR_C

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

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

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

$(MAIN): $(MAIN).o
      $(CC) $(CFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).c
      $(CC) $(CFLAGS) $(INC) -c $< -o $@

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

If you use the GNU compiler, use lines 4 and 5 and leave lines 1 and 2 as comments.

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 the C language 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.

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

The second argument is the NetCDF file name.

```
./sample_L3_DPR_C TEST ../data/GPM/DPR/GPMCOR_DPR_2603_M_L3S_D3M_08A.nc
lat=2.500000, lon=137.500000
precipRateESurface.mean[0][0][0][63][14]= 2.189663 (mm/hr)
lat=2.500000, lon=137.500000
precipRateESurface.mean[0][1][0][63][14]= 1.616606 (mm/hr)
lat=2.500000, lon=137.500000
precipRateESurface.mean[0][2][0][63][14]= 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 theversion 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 location

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.

```
#include ”TKheaders.h”
#include ”TK_2ADPR.h”

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

   /* declare Data struct */
   TKINFO granuleHandle2ADPR;
   L2ADPR_SWATHS L2ADPR;

   /* declare character */
   char job[256];
   char inputfile[256];

   strcpy(job, argv[1]);
   strcpy(inputfile, argv[2]);

  /* declare variables */
   int year, month, date, dayOfMon, hour, min;
   int i, j, k;
   int numOfScan;
   int status;
```

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".

```
   /* declare array */
   float lat[49], lon[49], precipESurf[49];
   float zFactorCor[49][176];

   /* NetCDF file open */
   status = TKopen(inputfile,"2ADPR"TKREAD,"NETCDF4",job,&granuleHandle2ADPR, 1);
   if(status != TK_SUCCESS) {
      fprintf(stderr, "error:file open error[%s] input2ADPR¥n", inputfile);
   }

   /* META data read */
   status = TKgetMetaInt(&granuleHandle2ADPR, "NS_SwathHeader",
                         "NumberScansGranule", &numOfScan);
   if(status != TK_SUCCESS) {
       fprintf(stderr, "error:file read error[%s]¥n", inputfile);
   }

  /* Scan data read */
   for (i=0; i<numOfScan; i++){
     status = TKreadScan( &granuleHandle2ADPR, &L2ADPR);

     /* Scantime Read */
     year      = L2ADPR.NS.ScanTime.Year;
     month     = L2ADPR.NS.ScanTime.Month;
     dayOfMon  = L2ADPR.NS.ScanTime.DayOfMonth;
     hour      = L2ADPR.NS.ScanTime.Hour;
     min       = L2ADPR.NS.ScanTime.Minute;

     for (j=0; j<49; j++){
/* Latitude and Longitude Read */
       lat[j]  = L2ADPR.NS.Latitude[j];
       lon[j]  = L2ADPR.NS.Longitude[j];

       /* precipRateESurface,zFactorCorrected Read */
       precipESurf[j] = L2ADPR.NS.SLV.precipRateESurface[j];
       for (k=0; k<176; k++){
         zFactorCor[j][k] = L2ADPR.NS.SLV.zFactorCorrected[j][k];
       }

       /* Print the value */
       if(i == 3946 && j == 19){
         printf("Scan=%d, Angle=%d¥n", i, j);
         printf("lat=%f, lon=%f¥n", lat[j], lon[j]);
         printf("precipESurf= %f (mm/hr)¥n", precipESurf[j]);
       }
     }
   }

   /* NetCDF File close*/
   status = TKclose(&granuleHandle2ADPR);

   return 0;
}
```

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

# 6. GPM/TRMM data loading with HDF library

Describes how to create a C 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.

```
#include <stdlib.h>
#include <string.h>
#include "hdf5.h"
int main(){

    hid_t  file_id, dataset_id;


    /* declare character */
    char inputfile[64]= "../data/GPM/DPR/
GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc";


    char dsetname[32]= "/FS/SLV/precipRateESurface";

    /* declare variables */
    int ret;


    /* declare array */
    float precipRateESurface[7988][49];

    /* NetCDF file open */
    file_id = H5Fopen(inputfile, H5F_ACC_RDONLY, H5P_DEFAULT);


    if(file_id < 0 ) {
        fprintf(stderr, "error:file open error[%s] ¥n", inputfile);
    }
    else
```

Always include when using the HDF library.

file_id and dataset_id are used as file and data identifiers, respectively.

inputfile sets the name of the NetCDF file to be read.

dsetname sets the name of the data to be read from the NetCDF file.

Defines an area to store read data.

NetCDF file open
inputfile: NetCDF file name
H5F_ACC_RDONLY: Access designation (read only)
H5P_DEFAULT: File access property list (normally H5P_DEFAULT is used)

If the open fails, file_id is set to a negative value. If successful, file_id is used in H5Dopen.

Open Data Sets
file_id: Specify the return value of H5Fopen.
dsetname: Name of the data to be read.
H5P_DEFAULT: File access property list (normally H5P_DEFAULT is used)

```
  {
    /* dataset open */
    dataset_id = H5Dopen(file_id, dsetname, H5P_DEFAULT);
```

If the open fails, a negative value is set to dataset_id. If successful, file_id is used in H5Dread.

```
    if(dataset_id < 0 ) {
      fprintf(stderr, "error:file open error[%s] ¥n", inputfile);
    }
    else
    {
```

data loading
dataset_id: Specify the return value of H5Dopen.
H5T_NATIVE_FLOAT: Memory type ID (specifies data type)
H5S_ALL, : Memory space ID (specifies H5S_ALL)
H5S_ALL, : File space ID
H5P_DEFAULT : Transfer property list (H5P_DEFAULT is specified)
precipRateESurface: specifies the area to store the read data.

```
      /* precipRateESurface read   */
      ret = H5Dread(dataset_id,H5T_NATIVE_FLOAT,H5S_ALL,H5S_ALL, H5P_DEFAULT,
precipRateESurface);
```

If the read fails, a negative value is set to ret.

```
      if(ret < 0) {
        fprintf(stderr, "error:Dataset read error[%s]¥n", inputfile);
      }
     else
     {
        /* file,dataset print */
        printf("file name= %s ¥n", inputfile);
        printf("data set name= %s ¥n", dsetname);
        /* precipRateESurface print */
        printf("precipRateESurface[5210][43])= %f (mm/hr)¥n",
precipRateESurface[5210][43]);
      }
```

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

Data Set Close
dataset_id: Specify the value obtained by H5Dopen.

```
      /* detaset close   */
      ret = H5Dclose(dataset_id);
```

Close NetCDF files
file_id: Specify the value obtained by H5Fopen.

```
    }
    /* NetCDF file close   */
    ret = H5Fclose(file_id);
  }
  return 0;
}
```

### 6.1.2 コンパイル方法

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

```
CC=icx
CFLAGS=-O0 -g -mcmodel=medium -fPIC

#CC=gcc
#CFLAGS=-g -mcmodel=medium -fpic -Wall

MAIN=./sample_HDF5_L2_DPR_C

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

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

LIBES = -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -lnetcdf

$(MAIN): $(MAIN).o
      $(CC) $(CFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).c
      $(CC) $(CFLAGS) $(INC) -c $< -o $@

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

If the Intel compiler is used as the compiler, lines 1 and 2 should be used and lines 4 and 5 should be comments.

Name of the program in 6.1.1.

The path to the NetCDF include directory

The path to the NetCDF library directory

### 6.1.3 Execution Results

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

```
$ ./sample_HDF5_L2_DPR_C
file name= ../data/GPM/DPR/GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc
data set name= /FS/SLV/precipRateESurface
precipRateESurface[5210][43]= 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.

```
#include <stdlib.h>
#include <string.h>
#include "hdf5.h"
int main(){

   hid_t  file_id, dataset_id;


   /* declare character */
   char inputfile[64]= "../data/GPM/DPR/GPMCOR_DPR_2603_M_L3S_D3M_08A.nc";



   char dsetname[64]= "/FS/G1/precipRateESurface/mean";

   /* declare variables */
   int ret;
   float  lat,lon;
   /* declare array */
   float precipRateESurface[3][3][3][72][28];



   /* NetCDF file open */
   file_id = H5Fopen(inputfile, H5F_ACC_RDONLY, H5P_DEFAULT);



   if(file_id < 0 ) {
     fprintf(stderr, "error:file open error[%s] ¥n", inputfile);
   }
   else{



     /* dataset open */
     dataset_id = H5Dopen(file_id, dsetname, H5P_DEFAULT);



     if(file_id < 0 ) {
       fprintf(stderr, "error:file open error[%s] ¥n", inputfile);
     }
```

Always include when using the HDF library.

file_id and dataset_id are used as file and data identifiers, respectively.

inputfile sets the name of the NetCDF file to be read.

dsetname sets the name of the data to be read from the NetCDF file.

NetCDF file open
inputfile: NetCDF file name
H5F_ACC_RDONLY: Access designation (read only)
H5P_DEFAULT: File access property list (normally H5P_DEFAULT is used)

If the open fails, file_id is set to a negative value. If successful, file_id is used in H5Dopen.

Open Data Sets
file_id: Specify the return value of H5Fopen.
dsetname: Name of the data to be read.
H5P_DEFAULT: File access property list (normally H5P_DEFAULT is used)

If the open fails, a negative value is set to dataset_id. If successful, file_id is used in H5Dread.

```
     else{
       /* precipRateESurface read   */
       ret = H5Dread(dataset_id,H5T_NATIVE_FLOAT,H5S_ALL,H5S_ALL, H5P_DEFAULT,
precipRateESurface);

       if(ret < 0) {
         fprintf(stderr, "error:Dataset read error[%s]¥n", inputfile);
       }
      else{
         /* file,dataset print */
         printf("File name= %s ¥n", inputfile);
         printf("Dataset name= %s ¥n", dsetname);

        /* lat/lon print */
        lat = (140.0/28.0) * 14 - 70.0 + (140.0/28.0/2);
        lon = (360.0/72.0) * 63 - 180.0 + (360.0/72.0/2);
        printf("lat=%f, lon=%f¥n",lat,lon);

         /* precipRateESurface print */
         printf("precipRateESurface.mean[0][0][0][63][14])= %f ¥n",
precipRateESurface[0][0][0][63][14]);
         printf("precipRateESurface.mean[0][1][0][63][14]= %f ¥n",
precipRateESurface[0][1][0][63][14]);
       }

      /* detaset close   */
       ret = H5Dclose(dataset_id);
     }

     /* NetCDF file close   */
     ret = H5Fclose(file_id);
   }
   return 0;
}
```

data loading
dataset_id: Specify the return value of H5Dopen.
H5T_NATIVE_FLOAT: Memory type ID (specifies data type)
H5S_ALL, : Memory space ID (specifies H5S_ALL)
H5S_ALL, : File space ID
H5P_DEFAULT : Transfer property list (H5P_DEFAULT is specified)
precipRateESurface: specifies the area to store the read data.

If the read fails, a negative value is set to ret.

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

Data Set Close
dataset_id: Specify the value obtained by H5Dopen.

Close NetCDF files
file_id: Specify the value obtained by H5Fopen.

### 6.2.2 Compilation Method

The following is an example of makefil e used at compile time.

```
CC=icx
CFLAGS=-O0 -g -mcmodel=medium -fPIC

#CC=gcc
#CFLAGS=-g -mcmodel=medium -fpic -Wall

MAIN=./sample_HDF5_L3_DPR_C

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

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

LIBES = -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -lnetcdf

$(MAIN): $(MAIN).o
      $(CC) $(CFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).c
      $(CC) $(CFLAGS) $(INC) -c $< -o $@

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

If the Intel compiler is used as the compiler, lines 1 and 2 should be used and lines 4 and 5 should be comments.

The name of the program in 6.2.1.

The path to the NetCDF include directory

The path to the NetCDF library directory

### 6.2.3 Execution Results

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

```
./sample_HDF5_L3_DPR_C
file name= ../data/GPM/DPR/GPMCOR_DPR_2603_M_L3S_D3M_08A.nc
data set name= /FS/G1/precipRateESurface/mean
lat=2.500000, lon=137.500000
precipRateESurface.mean[0][0][0][63][14]= 2.189663
precipRateESurface.mean[0][1][0][63][14]= 1.616606
```

# 7. h5dump to read GPM/TRMM data

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 C program to read that binary file.

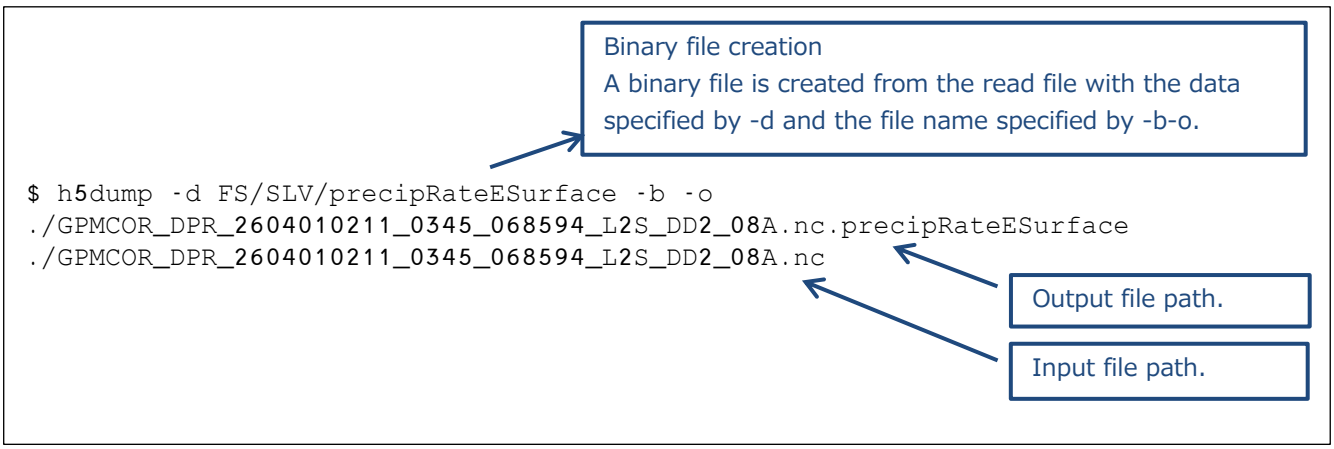
Explanations in red describe sample programs.

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.

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

int main(){

    /* declare variables */
    float precipRateESurface[7988][49];
    FILE  *fp;

    /* binary file declare */
    char inputfile[128] =”../data/GPM/DPR/
GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc.precipRateESurface”;


    /* binary file open */
    fp = fopen(inputfile, ”rb”);
    if(fp == NULL ) {
      printf(”error:file open error[%s] ¥n”, inputfile);
    }
    else
    {
      /* file read */
      fread(precipRateESurface, sizeof(precipRateESurface), 1, fp);

      /* precipRateESurface print */
      printf(”filename=%s¥n”,inputfile);
      printf(”precipRateESurface[5210][43])= %f (mm/hr)¥n”,
precipRateESurface[5210][43]);
      printf(”precipRateESurface[5210][44]= %f (mm/hr)¥n”,
precipRateESurface[5210][44]);

      /* binary file close   */
      fclose(fp);
    }
    return 0;
}
```

Defines the area where data is to be

The binary file created in 7.1.1 is specified.

Open binary file
The second argument should be "rb" (binary).

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.

```
CC=icx
CFLAGS=-O0 -g -mcmodel=medium -fPIC

#CC=gcc
#CFLAGS=-g -mcmodel=medium -fpic -Wall

MAIN=./sample_h5dump_L2_C

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

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

LIBES = -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -lnetcdf

$(MAIN): $(MAIN).o
      $(CC) $(CFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).c
      $(CC) $(CFLAGS) $(INC) -c $< -o $@

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

If the Intel compiler is used as the compiler, lines 1 and 2 should be used and lines 4 and 5 should be comments.

The name of the program in 7.1.2.

The path to the NetCDF include directory

The path to the NetCDF library directory

### 7.1.4 Execution results

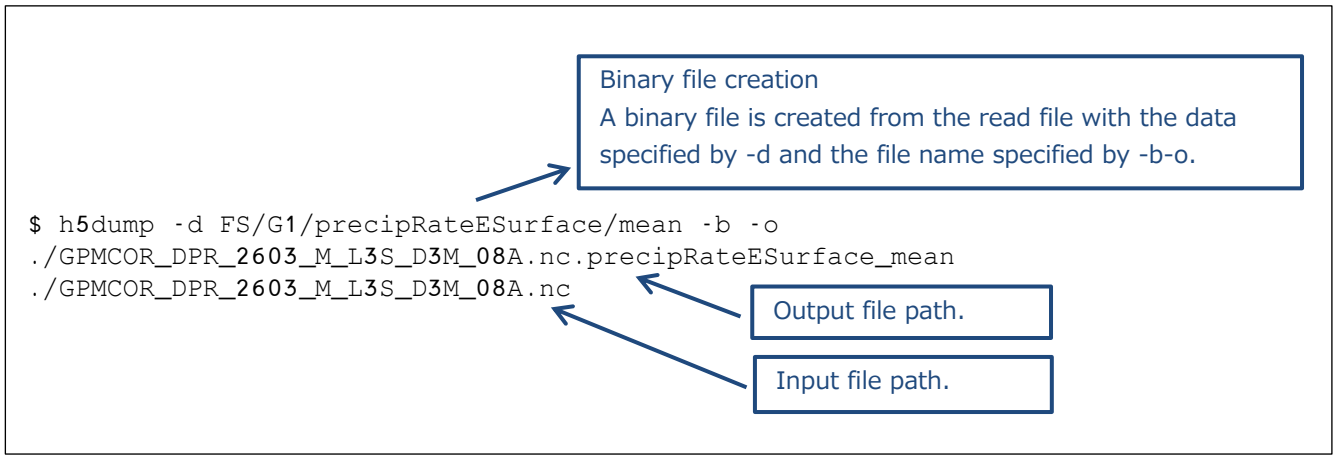
The following are the results of executing the program described in 7.1.2.

```
$ ./sample_h5dump_L2_C
filename=../data/GPM/DPR/GPMCOR_DPR_2604010211_0345_068594_L2S_DD2_08A.nc.precipRateESurfa
ce
precipRateESurface[5210][43]= 2.261208 (mm/hr)
precipRateESurface[5210][44]= 2.951939 (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.

```
#include <stdio.h>
#include <string.h>
int main(){

    /* declare variables */
    float precipRateESurface[3][3][3][72][28];
    FILE  *fp;
    int   i, j, k, m, n, size;
    float  lat,lon;
    /* binary file name */
    char inputfile[128]
="../data/GPM/DPR/GPMCOR_DPR_2603_M_L3S_D3M_08A.nc.precipRateESurface_mean";

    for(i=0; i<3; i++){
       for(j=0; j<3; j++){
           for(k=0; k<3; k++){
               for(m=0; m<72; m++){
                   for(n=0; n<28; n++){
                       precipRateESurface[i][j][k][m][n]=0;
                   }
               }
           }
       }
   }


    /* binary file open */
    fp = fopen(inputfile, "rb");
    if(fp == NULL ) {
      printf("error:file open error[%s] ¥n", inputfile);
    }
    else
    {
      /* file read */
      size = fread(precipRateESurface, sizeof(precipRateESurface), 1, fp);

      /* precipRateESurface print */
          printf("filename=%s¥n",inputfile);
      lat = (140.0/28.0) * 14 - 70.0 + (140.0/28.0/2);
      lon = (360.0/72.0) * 63 - 180.0 + (360.0/72.0/2);
      printf("lat=%f, lon=%f¥n",lat,lon);
      printf("precipRateESurface.mean[0][0][0][63][14]= %f ¥n",
precipRateESurface[0][0][0][63][14]);
      printf("precipRateESurface.mean[0][1][0][63][14]= %f ¥n",
precipRateESurface[0][1][0][63][14]);

      /* binary file close   */
      fclose(fp);
    }
    return 0;
 }
```

Defines the area where data is to be

The binary file created in 7.2.1 is specified.

Open binary file
The second argument should be "rb" (binary).

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.

```
CC=icx
CFLAGS=-O0 -g -mcmodel=medium -fPIC

#CC=gcc
#CFLAGS=-g -mcmodel=medium -fpic -Wall

MAIN=./sample_h5dump_L3_C

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

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

LIBES = -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -lnetcdf

$(MAIN): $(MAIN).o
      $(CC) $(CFLAGS) -o $@ $^ $(LIB) $(LIBES)

$(MAIN).o: $(MAIN).c
      $(CC) $(CFLAGS) $(INC) -c $< -o $@

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

If the Intel compiler is used as the compiler, lines 1 and 2 should be used and lines 4 and 5 should be comments.

The name of the program in 7.2.2.

The path to the NetCDF include directory

The path to the NetCDF library directory

### 7.2.4 Execution Results

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

```
$ ./sample_h5dump_L3_C
read size=1
filename=../data/GPM/DPR/GPMCOR_DPR_2603_M_L3S_D3M_08A.nc.precipRateESurface_mean
lat=2.500000, lon=137.500000
precipRateESurface.mean[0][0][0][63][14]= 2.189663
precipRateESurface.mean[0][1][0][63][14]= 1.616606
```

# 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.<br>Table 1.2 Sample code operation check table was added.<br>4. installation of library tools: Table 4.2 Product bar version and PPS Toolkit (TKIO).<br>In Table 4.3 Operating Environment, where it says tkio-3.70.7<br>Changed to tkio-x.xx.x<br>4.2.4Edit configuration file: Change "tkio-3.70.7" to "tkio-x.xx.x". | |
| 3 | Mar 15, 2018 | 3. Related documents and sample programs available: Table 3.1 sample program list added. | |
| 4 | Jan 25, 2019 | 1.-3. Correction due to addition of TRMM and renewal of GPM site<br>4. Table 4.2 Product Version and TKIO Compatible Versions Corrected to list the product version and TKIO compatible version for each product.<br>5. added a list of algorithm IDs and changed the sample programs to include one per level | |
| 5 | Dec 24, 2021 | 1. modified to GSMaP product version 5 and GPM/TRMM product version 7.<br>3. revised availability of related documentation and sample programs<br>Correction of URL for TKIO download<br>3.1 - 5.1: 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.<br>5. modified V7 changes in the program | |
| 6 | Jan 21, 2022 | 5.-7. Corrections due to modification of sample programs (review of display position, addition of lat/lon to display items, etc.) | |
| 7 | Jun 1, 2026 | Sample data updated to V8<br>Corrected code description to match V8 and NetCDF | |