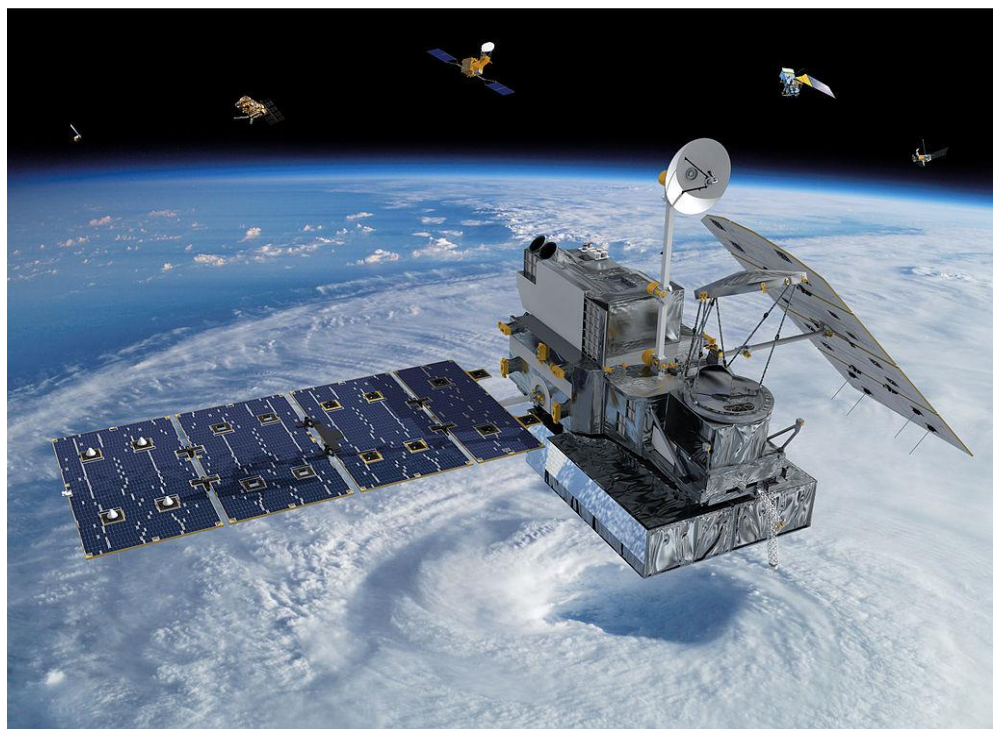


# GPM/TRMM Data reading program guide (C language version)



2022/01/21

6th 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 07 for GPM/TRMM and with product version 05 for GSMaP.

## Table of Contents

1. Introduction.....	3
2. How to obtain GPM/TRMM data .....	5
3. how to obtain related documents and sample programs.....	8
4. installation of library tools.....	10
4.1 Installation of HDF5 .....	11
4.2 Installation of PPS Toolkit (TKIO) .....	12
GPM/TRMM data reading with PPS Toolkit (TKIO) .....	15
5.1 L1 data reading .....	17
5.2 L2 data reading .....	20
5.3 L3 Data Read.....	23
5.4 Reading GSMP_HDF5 Data.....	26
5.5 About the version of the PPS Toolkit (TKIO) .....	28
6. GPM/TRMM data loading in HDF library .....	30
6.1 Loading L2DPR Data .....	30
6.2 Loading L3DPR Data .....	33
6.3 GSMP_HDF5 Data Read .....	36
7. GPM/TRMM data read by h5dump .....	39
7.1 L2 data reading .....	39
7.2 L3 Data Read.....	42

## Introduction

This document explains how to read in GPM/TRMM data using the C language.

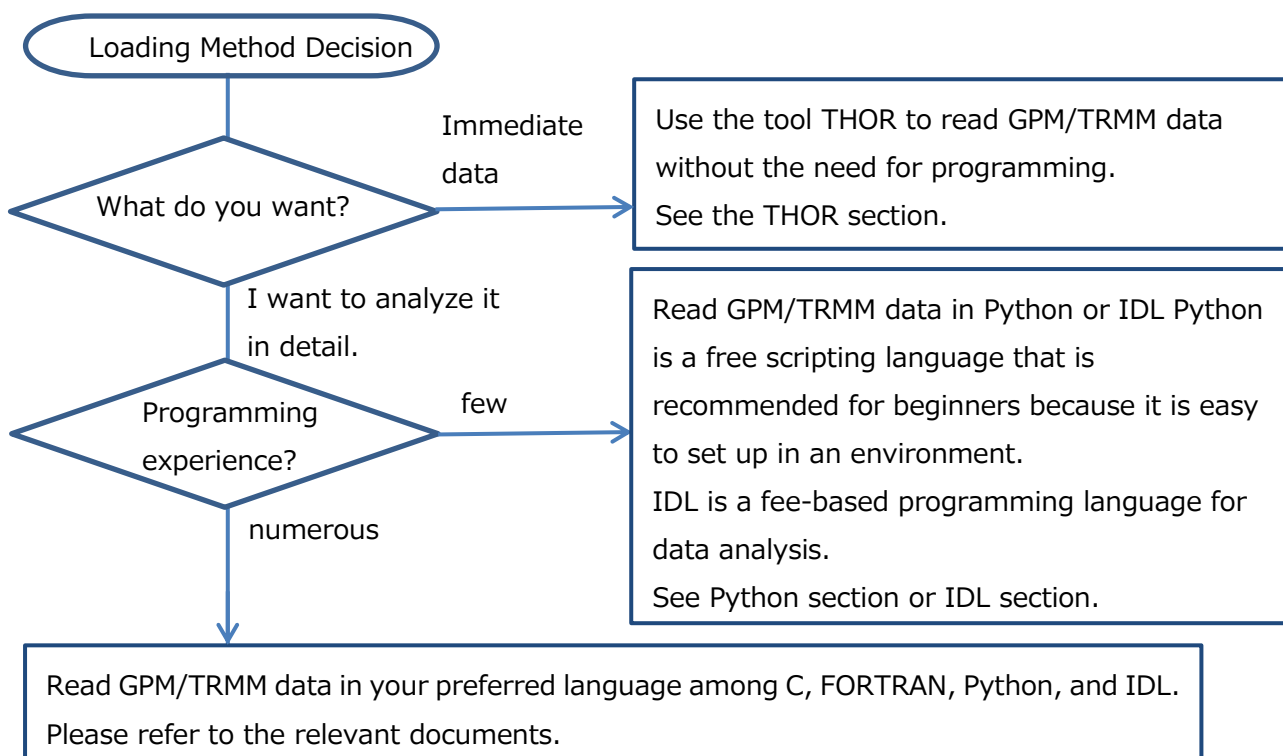
The GPM and TRMM formats have been unified since version 06 products (equivalent to TRMM version 8), and the latest algorithm is version 07 (equivalent to TRMM version 9). The latest algorithm is version 07 (equivalent to TRMM version 9), which can be read in the same way in this sample program.

In addition to 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	Use IDL	GPM/TRMM Data Loading Program Guide (IDL version)	
3	Use C	GPM/TRMM Data Loading Program Guide (C language version)	
4	Using FORTRAN	GPM/TRMM Data Loading Program Guide (FORTRAN Edition)	
5	Using Python	GPM/TRMM data reading program guide (Python version)	



**Table 1.2 Sample Program Operation Check Table**

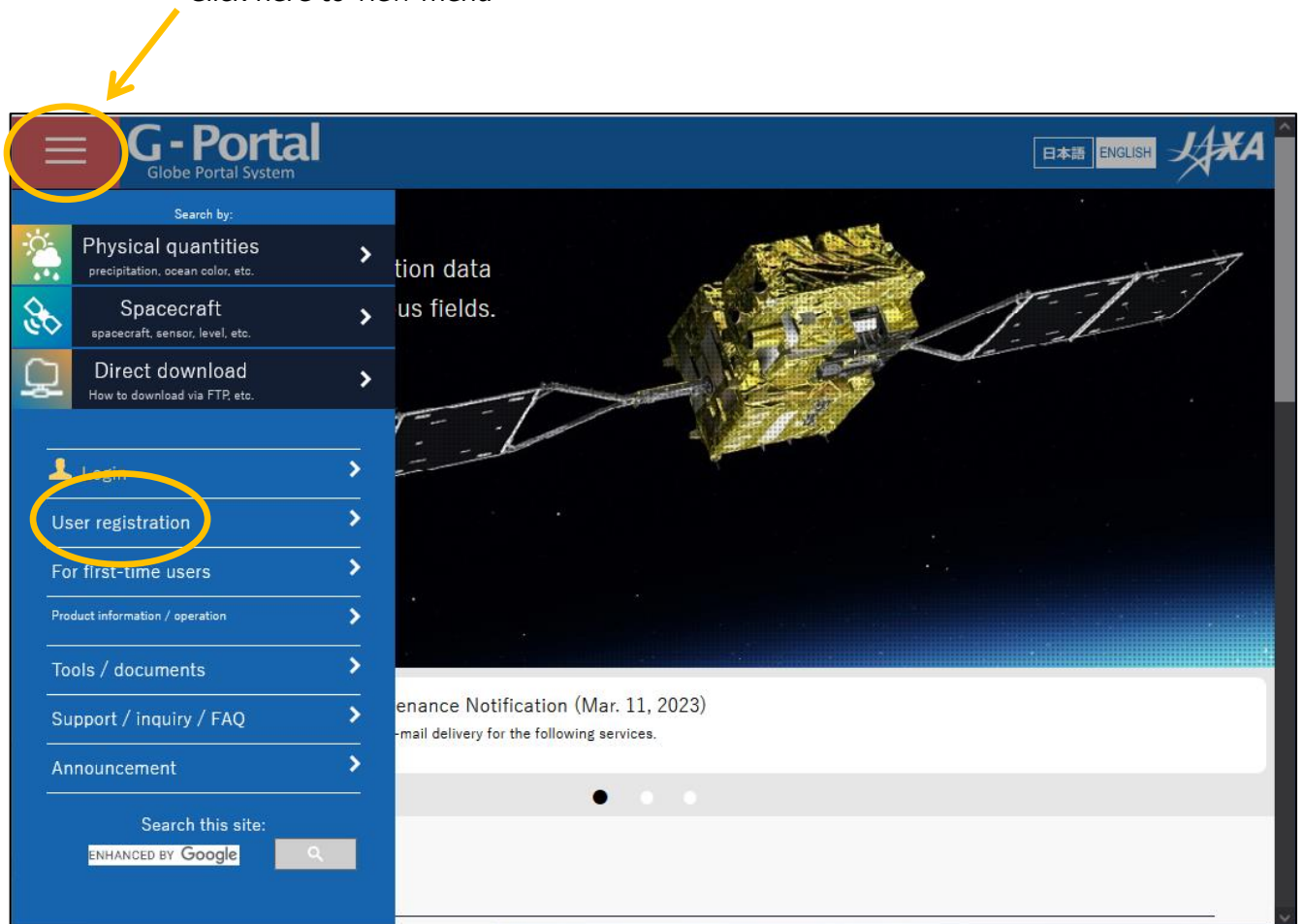
	sample program	Linux	Windows	remarks
1	c	○	-	
2	FORTRAN	○	-	
3	Python	○	○	
4	IDL	○	○	

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

## 2. how to obtain GPM/TRMM data

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

Click here to view menu



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

The screenshot shows the G-Portal user registration interface. At the top, there is a navigation bar with the G-Portal logo, the text "Globe Portal System", and language options for "日本語" and "ENGLISH", along with the JAXA logo. Below the navigation bar is a progress indicator with five steps: 1. Terms of Use, 2. Enter registration information, 3. Confirm registration information, 4. Temporary registration completed, and 5. Registration completed. The current step is Step 1, "Terms of Use".

The main content area is titled "User Registration STEP1/5: G-Portal Terms of Use". Below the title, a message states: "You need to register as a user to download products from G-Portal. Please read and accept the following terms and proceed to the next step:". Below this message is a scrollable box containing the "Terms of Use" text. The text includes a heading "G-Portal Terms of Use" and a paragraph stating: "G-Portal is a free service providing data of spaceborne sensors that Japan Aerospace Exploration Agency (JAXA) has developed/involved. This Terms of Use states the terms and conditions under which you may use G-Portal. [JAXA Site Policy](#) is applied to the matter which is not specified in this Terms of Use. Please read carefully and make sure you accept this Terms of Use before using G-Portal. In order to use G-Portal, the user must agree to this Terms of Use. You can accept the Terms by clicking to agree to this Terms of Use, where this option is made available to the user by JAXA; or by actually using the services. In the latter case, the user understands and agrees that JAXA will treat the user's use of G-Portal as acceptance of the Terms of Use from that point onwards."

Below the terms text, there is a section titled "1. User Registration" with the following text: "You need to create a user account to use G-Portal. Your user account and password will serve as your login information. The items required for G-Portal user registration are: a username, a valid e-mail address, the name of a user's affiliation, country or region of a user, and a user's purpose of use. For security reason, G-Portal requires you to use a valid e-mail address that identifies your educational or company affiliation (i.e., @jaxa.jp, @XX.edu, @companyname.com or @XX.org). If you use any e-mail address like Gmail, Yahoo, or any other free mail, you may not be able to complete your registration, or may not be able to receive e-mails from G-Portal."

At the bottom of the scrollable box, there is a checkbox labeled "I agree to the above terms of service". Below the scrollable box, there are two buttons: "I Agree - Continue" and "Do Not Agree".

You will be taken to the user registration screen.

**G-Portal**  
Globe Portal System

日本語 ENGLISH JAXA

1 Terms of Use 2 Enter registration information 3 Confirm registration information 4 Temporary registration completed 5 Registration completed

### User Registration STEP2/5: G-Portal Registering User Information

Please complete all the following items and press "Confirm Registration Information":

User account (Required):

Password (Required) ⓘ :

Password (reconfirm) (Required):

Name (Required):

Email address (Required) ⓘ :

Email address (reconfirm) (Required):

Organization:

Department:

Country:

Language (Required) ⓘ :  Japanese  English

Analysis

Algorithm Development

Data Validation

Applied Research

Education

Calibration

Order-made

Other

Purpose (Required):

Email Delivery Preference (Required) ⓘ :  By order  By preparation

**\*Handling of email addresses**

On this site, we strongly recommend using your corporate or institutional mail address (such as @jaxa.jp), to ensure you receive URL information of ordered products and user registration. If you do not receive such email, or if you receive an unexpected email, please contact the Support Desk. If you use a free email address (like @gmail.com, icloud.com) or private email, our email may not reach you.

**\*Be aware of phishing scams**

Avoid filling out forms contained in email messages that request personal information. We will never send any email requesting your user account or password.

Next

Cancel

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

### 3. how to obtain related documents and sample programs

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

Documentation for GPM data use includes

- GPM Data Application Handbook
- file naming convention

The screenshot shows the 'Archives' section of the GPM website. The breadcrumb trail is 'Top > Archives > TRMM/GPM V07'. A navigation bar contains buttons for 'TRMM/GPM V07', 'TRMM/GPM V06', 'TRMM/GPM V06X', 'GPM/V05', 'TRMMV7A', 'GSMaP', 'References', and 'Others'. The main heading is 'TRMM/GPM Products (Version07)'. Below this, a text block states: 'The format of L2/L3 products for GPM (Version06) and TRMM (corresponding to V8) has been integrated and the latest algorithm is Version07 (TRMM corresponding to V9)'. A table follows, detailing product information for TRMM and GPM.

		TRMM	GPM	
	PR/DPR L1B	V07 (corresponded to V9)	V07	2014/03/08-current V07
	PR/DPR L2/L3	V07 (corresponded to V9)	V07	2014/03/08-current V07
	SLH	V07 (corresponded to V9)	V07	2014/03/08-current V07
NASA	PR/DPR comb.(CSH)	V07 (corresponded to V9)	V07	2022/05/09-current V07
	VIRS/TMI/GMI	V07 (corresponded to V9)	V07	2022/05/09-current V07

at 2022/05

The screenshot shows the 'Data Utilization' section of the GPM website. The breadcrumb trail is 'Top > Data Utilization'. The main heading is 'Data Download'. Below it, there is a link: 'GPM products "G-Portal Earth observation satellite data providing system"'. Further down, the heading 'Data Utilization' is shown, followed by three links: 'Data Utilization Handbook', 'Documents related to products are here', and 'Papers related to products are here'.

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



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

**Table 3.1 List of Sample Programs**

product	sample program	sample data
L1Ku	sample_L1_Ku_C.c	GPMCOR_KUR_2112070007_0140_044170_1BS_DUB_07A.h5
L2DPR	sample_L2_DPR_C.c	GPMCOR_DPR_2112070007_0140_044170_L2S_DD2_07A.h5
	sample_HDF5_L2_DPR_C.c	
	sample_h5dump_L2_C.c	
L3DPR	sample_L3_DPR_C.c	GPMCOR_DPR_1806_M_D3M_07X.EORC.h5
	sample_HDF5_L3_DPR_C.c	
	sample_h5dump_L3_C.c	
L2GMI	sample_L2_GMI_C.c	GPMCOR_GMI_1709252152_2324_020322_L2S_GL2_05A.h5
L3GMI	sample_L3_GMI_C.c	GPMCOR_GMI_1707_M_L3S_GL3_05A.h5
L2CMB	sample_L2_CMB_C.c	GPMCOR_CMB_1901082158_2331_027633_L2S_CL2_06A.h5
L3CMB	sample_L3_CMB_C.c	GPMCOR_CMB_1812_M_L3S_CL3_06A.h5
L1PR	sample_L1_PR_C.c	GPMTRM_KUR_0901311508_1640_063883_1BS_PU1_V07X_20210630.h5
L2PR	sample_L2_PR_C.c	GPMTRM_KUR_0901311508_1640_063883_L2S_PU2_V07X_20211129.h5
L3PR	sample_L3_PR_C.c	3A-MO.TRMM.PR.V9-20210117.19980101-S000000-E235959.01.TRMM700.HDF5
L2SLP	sample_L2_SLP_C.c	GPMCOR_KUR_1512282046_2218_010412_L2S_SLP_07X_20211018.EORC.h5
L3SLM	sample_L3_SLM_C.c	GPMCOR_KUR_1512_L3S_SLM_07X_20211019.EORC.h5
L3SLG	sample_L3_SLG_C.c	GPMCOR_DPR_2112010038_0211_044077_L3S_SLG_07A.h5
L2LHP	sample_L2_LHP_C.c	GPMTRM_KUR_1503312350_0121_098980_L2S_LHP_07X_20211018.EORC.h5
L3LHM	sample_L3_LHM_C.c	GPMTRM_KUR_1503_L3S_LHM_07X_20211019.EORC.h5
L3LHG	sample_L3_LHG_C.c	GPMTRM_KUR_1503312350_0121_098980_L3S_LHG_07X_20211028.EORC.h5
GSMaP	sample_GSMaP_HDF5_C.c	GPMMRG_MAP_2112010000_H_L3S_MCH_05A.h5
	sample_HDF5_GSMaP_C.c	

## 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, PPS TKIO	See Table 4.2
2	HDF5 Library	HDF5	
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	07	3.97.18	
L2DPR	07	3.97.18	
L3DPR	07	3.97.18	
L2GMI	05	3.80.26	
L3GMI	05	3.80.26	
L2CMB	06	3.90.0	
L3CMB	06	3.90.0	
L1PR	07	3.97.18	
L2PR	07	3.97.18	
L3PR	07	3.97.18	
L2SLP	07	3.97.18	
L3SLM	07	3.97.18	
L3SLG	07	3.97.18	
L2LHP	07	3.97.18	
L3LHM	07	3.97.18	
L3LHG	07	3.97.18	
GSMaP	04	3.80.10	

Note: PPS Toolkit (TKIO) is basically upward compatible, but may not load properly in some cases. In that case, please refer to "5.5 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 ES-2665 2.4GHz
OS	Red Hat Enterprise Linux Server release 6.4
C compiler	gcc 4.4.7,icc 14.0.1
HDF5	Hdf5-1.8.9
PPS TKIO	tkio-x.xx.x

## 4.1 Installation of HDF5

### 4.1.1 Download

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

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

### 4.1.2 Decompression

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

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

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

```
$ cd hdf5-1.8.9
```

### 4.1.3 Compilation and Installation

Execute the following commands in order to compile and install

--prefix= specifies the directory to install in.

\*In this example, the version of hdf5 is 1.8.9, so hdf5\_1.8.9 is used.

Replace the version letter part with the version actually used.

<If you do not use the FORTRAN library in HDF5

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

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

```
$ make
```

```
$ make install
```

<If you use the FORTRAN library in HDF5

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

```
  --with-szlib=/home/user1/util/szip_2.1 --enable-fortran FC=ifort
```

```
$ make
```

```
$ make install
```

## 4.2 Installation of PPS Toolkit (TKIO)

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

### 4.2.1 Download

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

### 4.2.2 Decompression

Create an appropriate working directory, move the downloaded files into it, and extract the compressed files.

You can decompress it with the following command.

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

### 4.2.3 Checking Assumptions

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

#### Installation of libxml2 library

Check docs/tkioINSTALL.txt for the version you need!

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

```
make
```

```
make install
```

#### Installation of zlib library

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

```
make
```

```
make install
```

#### Installation of jpeg library

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

```
make
```

```
make install
```

```
make install-lib
```

#### Installation of hdf4 library

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

make

make install

Installation of hdf5 library

. /configure --prefix=[install-DIR] --enable-fortran --with-zlib=[zlib install-DIR]

--with-szlib=[szip install-DIR] CC=icc CXX=icpc FC=ifort

make

make install

#### 4.2.4 Editing the Preferences File

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

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

#### 4.2.5 Reading the Preferences File

The following command reads the preferences file.

```
$ source Preferences file name
```

#### 4.2.6 Compilation

Execute the following commands to compile.

```
$ ./INSTALL.pl compileJAVA
$ ./INSTALL.pl buildRW
$ ./INSTALL.pl compileRW
```

## 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 HDF5 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 HDF5 file name of L1Ku are used as arguments, and the following data are read from the file specified as the argument: date and time information, latitude and longitude information, echoPower, and noisePower.

```

1:#include "TKheaders.h"
2:#include "TK_1BKu.h"
3:
4:#define NRAY 49 /* number of Anglebin in one scan */
5:#define NBIN 260 /* number of Rangebin in one Anglebin */
6:
7:int main(int argc, char *argv[]){
8:
9: /* declare Data struct */
10: TKINFO granuleHandle1BKu;
11: L1BKu_FS L1BKu;
12:
13: /* declare character */
14: char job[256];
15: char inputfile[256];
16:
17: strcpy(job, argv[1]);
18: strcpy(inputfile, argv[2]);
19:
20: /* declare variables */
21: int year, month, dayOfMon, hour, min;
22: int i, j, k;
23: int numOfScan;
24: int status;
25:
26: /* declare array */
27: float lat[NRAY], lon[NRAY];
28: short noisePower[NRAY], echoPower[NRAY][NBIN];
29:
30: /* HDF file open */
31: status = TKopen(inputfile, "1BKu", TKREAD, "HDF5", job, &granuleHandle1BKu, 1);
32: if(status != TK_SUCCESS) {
33: fprintf(stderr, "error:file open error[%s] input1BKu¥n", inputfile);
34: }
35:

```

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 HDF5 files and is used when using the PPS Toolkit.

L1BKu is the structure of the HDF5 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 HDF5 file  
inputfile:HDF5 file name (string specified by argument)  
1BKu:Algorithm ID  
TKREAD:Read designation  
HDF5: 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 HDF5 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

```

36: /* META data read */
37: status = TKgetMetaInt(&granuleHandle1BKu, "SwathHeader",
38: "NumberScansGranule", &numOfScan);
39: if(status != TK_SUCCESS) {
40: fprintf(stderr, "error:file read error[%s]\n", inputfile);
41: }
42:

```

Loop for number of scans

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

```

43: /* Scan data read */
44: for (i=0; i<numOfScan; i++){
45: status = TKreadScan( &granuleHandle1BKu, &L1BKu);
46:
47: /* Scantime Read */
48: year = L1BKu.ScanTime;
49: month = L1BKu.ScanTime.Month;
50: dayOfMon = L1BKu.ScanTime.DayOfMonth;
51: hour = L1BKu.ScanTime;
52: min = L1BKu.ScanTime.Minute;
53:

```

Read scan date and time

```

54: for (j=0; j<NRAY; j++){
55:     /* Latitude and Longitude Read */
56: lat[j] = L1BKu.Latitude[j];
57: lon[j] = L1BKu.Longitude[j];
58:

```

Latitude and longitude

```

59:     /* noisePower Read */
60: noisePower[j] = L1BKu.Receiver.noisePower[j];
61:

```

Reading noisePower

```

62:     /* echoPower Read */
63: for (k=0; k<NBIN; k++){
64: echoPower[j][k] = L1BKu.Receiver.echoPower[j][k];
65:     }
66:

```

echoPower loading

```

67:     /* Print the value */
68:     if(i == 3946 && j == 19){
69: printf("Scan=%d, Angle=%d\n", i, j);
70: printf("lat=%f, lon=%f\n", lat[j], lon[j]);
71: printf("echoPower[259]= %d \n", echoPower[j][259]);
72: printf("noisePower= %d \n", noisePower[j]);
73:     }
74: }
75: }
76:

```

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

Close HDF files  
 &granuleHandle1BKu: File pointer (specifies TKINFO structure)

```

77: /* HDF File close*/
78: status = TKclose(&granuleHandle1BKu);
79:
80: return 0;
81: }

```

### 5.1.2 Compilation Method

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

```

1:#cc=icc
2:#CFLAGS=-g -shared-intel -mmodel=medium
3:
4:cc=gcc
5:CFLAGS=-g -mmodel=medium -fpic -Wall
6:
7:MAIN=. /sample_L1_Ku_C
8:
9:INC = -I$(HDF5_INC) %
10:  -I$(TKIO)/inc/ccode %c
11:  -I$(SZIP_INC)
12:
13:LIB = -L$(HDF5_LIB) %1}LIB %
14:  -L$(TKIO)/lib %c
15:  -L$(SZIP_LIB)
16:
17:LIBES = -ltkselect %
18:  -ltkchdf5algs -ltkchdf5 %c
19:  -ltk -lm -lmfhdf -ldf -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -lsz -ltirpc
20:
21:$(MAIN): $(MAIN).o
22:  $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
23:
24:$(MAIN).o: $(MAIN).c
25:  $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
26:
27:clean:
28:  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 5.1.1.

The path to the HDF5 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 "bb" "GPMCOR_KUR_2112070007_0140_044170_1BS_DUB_07A.h5"
Scan=3946, Angle=19
lat=64.878418, lon=121.662895
echoPower[260]= -29999
noisePower= -11092$.

```

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

The second argument is the HDF5 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 HDF5 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.

```

1:#include "TKheaders.h"
2:#include "TK_2ADPR.h"
3:
4:#define NRAY 49 /* number of Anglebin in one scan */
5:#define NBIN 176 /* number of Rangebin in one Anglebin */
6:
7:int main(int argc, char *argv[]){
8:
9: /* declare Data struct */
10: TKINFO granuleHandle2ADPR;
11: L2ADPR_SWATHS L2ADPR;
12:
13: /* declare character */
14: char job[256];
15: char inputfile[256];
16:
17: strcpy(job, argv[1]);
18: strcpy(inputfile, argv[2]);
19:
20:
21: /* declare variables */
22: int year, month, date, dayOfMon, hour, min;
23: int i, j, k;
24: int numOfScan;
25: int status;
26:
27: /* declare array */
28: float lat[NRAY], lon[NRAY], precipESurf[NRAY];
29: float precipWater[NRAY][NBIN];
30:
31: /* HDF file open */
32: status = TKopen(inputfile, "2ADPR", TKREAD, "HDF5", job, &granuleHandle2ADPR, 1);
33: if(status != TK_SUCCESS) {
34: fprintf(stderr, "error:file open error[%s] input2ADPR¥n", inputfile);
35: }
36:

```

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 HDF5 files and is used when using the PPS

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

job is used for the argument specified in TKopen.

```

37: /* META data read */
38: status = TKgetMetaInt(&granuleHandle2ADPR, "FS_SwathHeader",
39: "NumberScansGranule", &numOfScan);
40: if(status != TK_SUCCESS) {
41: fprintf(stderr, "error:file read error[%s]\n", inputfile);
42: }
43:
44: /* Scan data read */
45: for (i=0; i<numOfScan; i++){
46: status = TKreadScan( &granuleHandle2ADPR, &L2ADPR);
47:
48: /* Scantime Read */
49: year = L2ADPR.FS.ScanTime.Year;
50: month = L2ADPR.FS.ScanTime.Month;
51: dayOfMon = L2ADPR.FS.ScanTime.DayOfMonth;
52: hour = L2ADPR.FS.ScanTime;
53: min = L2ADPR.FS.ScanTime.Minute;
54:
55: for (j=0; j<NRAY; j++){
56:     /* Latitude and Longitude Read */
57: lat[j] = L2ADPR.FS.Latitude[j];
58: lon[j] = L2ADPR.FS.Longitude[j];
59:
60:     /* precipRateESurface Read */
61: precipESurf[j] = L2ADPR.FS.SLV.precipRateESurface[j];
62:     for (k=0; k<NBIN; k++){
63: precipWater[j][k] = L2ADPR.FS.SLV.precipWater[j][k];
64:     }
65:
66:     /* Print the value */
67:     if(i == 1491 && j == 43){
68: printf("Scan=%d, Angle=%d\n", i, j);
69: printf("lat=%f, lon=%f\n", lat[j], lon[j]);
70: printf("precipRateESurface= %f (mm/hr)\n", precipESurf[j]);
71: printf("precipWater[43][100]= %f (g/m~3)\n", precipWater[j][100]);
72:     }
73: }
74: }
75:
76: /* HDF File close*/
77: status = TKclose(&granuleHandle2ADPR);
78: return 0;
79: }

```

Metadata read (scan count readout)  
 &granuleHandle2ADPR: File pointer (specifies TKINFO structure)  
 FS\_SwathHeader": The name of the item in the HDF5 file that contains the data to be read.  
 Information is stored. It is necessary to check the item names beforehand in the "L1 Product Format Description" or PPS Viewer THOR.  
 NumberScansGranule": specifies the number of scans.  
 &numOfScan: Address to store the number of scans read

Loop for number of scans

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

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 1491st 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.

```

1:cc=icc
2:CFLAGS=-g -shared-intel -mcmmodel=medium
3:
4:#cc=gcc
5:#CFLAGS=-g -mcmmodel=medium -fpic -Wall
6:
7:MAIN=. /sample_L2_DPR_C
8:
9:INC = -I$(HDF5_INC) ¥
10:  -I$(TKIO)/inc/ccode ¥c
11:  -I$(SZIP_INC)
12:
13:LIB = -L$(HDF5_LIB) ¥1}LIB =
14:  -L$(TKIO)/lib ¥c
15:  -L$(SZIP_LIB)
16:
17:LIBES = -ltkselect ¥
18: -ltkchdf5algs -ltkchdf5 ¥
19:  -ltk -lm -lmfhdf -ldf -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -ltirpc
20:
21:$(MAIN): $(MAIN).o
22:  $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
23:
24:$(MAIN).o: $(MAIN).c
25:  $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
26:
27:clean:
28:  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 5.2.1.

The path to the HDF5 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

### 5.2.3 Execution results

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

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

The second argument is the HDF5 file name.

```

$ ./sample_L2_DPR_C "cc" "GPMCOR_DPR_2112070007_0140_044170_L2S_DD2_07A.h5"
scan=1491, Angle=43
lat=-19.811699, lon=30.270908
precipRateESurface= 3.003061 (mm/hr)
precipWater[43][100]= 0.039507 (g/m~3)

```

## 5.3 L3 data reading

### 5.3.1 Source Programs

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

```

1:#include "TKheaders.h"
2:#include "TK_3DPR.h"

3:
4:#define LTL 28 /* number of low resolution 5° grid intervals of latitude from 70°S to 70°N.
*/
5:#define LNL 72 /* number of low resolution 5° grid intervals of longitude from 180°W to 180°E.
*/
6:#define CHN 3 /* number of channels: KuFS, KaMS, KaHS, DPRMS, KuMS, KaFS, DPRFS */
7:#define RT 3 /* number of rain types: all, stratiform, convective */
8:#define ST 3 /* number of surface types: all, ocean, land */
9:
10:int main(int argc, char *argv[]){
11:
12: /* declare Data struct */
13: TKINFO granuleHandle3DPR;
14: L3DPR_GRIDS L3DPR;
15:
16: /* declare character */
17: char job[256];
18: char inputfile[256];
19:
20: strcpy(job, argv[1]);
21: strcpy(inputfile, argv[2]);
22:
23: /* declare variables */
24: int i, j, k, l, m, n;
25: int status;
26: float lat,lon;
27: /* declare array */
28: float precipESurf[ST][RT][CHN][LHL][LTL];
29:
30: /* HDF file open */
31: status = TKopen(inputfile, "3DPR", TKREAD, "HDF5", job, &granuleHandle3DPR, 1);
32: if(status != TK_SUCCESS) {
33: fprintf(stderr, "error:file open error[%s] input3DPR¥n", inputfile);
34: }
35:

```

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 HDF5 files and is used when using the PPS Toolkit.

L3DPR is the structure of an HDF5 file, which contains data for one scan.

Open HDF5 file  
inputfile:HDF5 file name (string specified by argument)  
3DPR: Algorithm ID  
TKREAD:Read designation  
HDF5: Format type  
job:job name (string specified by argument)  
&granuleHandle2BCMB: File pointer (specifies TKINFO structure)  
1: Internal compression of files (specify 1)

```

36: /* Grid data read */
37: status = TKreadGridMulti( &granuleHandle3DPR, &L3DPR, "L3DPR_FS");
38:
39: /* precipRateESurface.mean data read */
40: for (j=0; j<ST; j++){
41:     for (k=0; k<RT; k++){
42:         for (l=0; l<CHN; l++){
43:             for (m=0; m<LNL; m++){
44:                 for (n=0; n<LTL; n++){
45:                     precipESurf[j][k][l][m][n] = L3DPR.G1.precipRateESurface.mean[j][k][l][m][n];
46:                     if( m==63 && n==14 && l=0 && j=0 ) {
47:                         lat = (140.0/28.0) * n - 70.0 + (140.0/28.0/2);
48:                         lon = (360.0/72.0) * m - 180.0 + (360.0/72.0/2);
49:                         printf("lat=%f, lon=%f\n",lat,lon);
50:                         printf("precipRateESurface.mean[%d][%d][%d][%d][%d]= %f (mm/hr)\n",
51:                             j, k, l, m, n, precipESurf[j][k][l][m][n]);
52:                     }
53:                 }
54:             }
55:         }
56:     }
57: }
58:
59: /* HDF File close*/
60: status = TKclose(&granuleHandle3DPR);
61:
62: return 0;
63: }

```

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 HDF files  
 &granuleHandle3DPR: File pointer (specifies TKINFO structure)



### 5.3.2 Compilation Method

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

```

1:cc=icc
2:CFLAGS=-g -shared-intel -mmodel=medium
3:
4:#cc=gcc
5:#CFLAGS=-g -mmodel=medium -fpic -Wall
6:
7:MAIN=. /sample_L3_DPR_C
8:
9:INC = -I$(HDF5_INC) %
10:  -I$(TKIO)/inc/ccode %c
11:  -I$(SZIP_INC)
12:
13:LIB = -L$(HDF5_LIB) %1}LIB =
14:  -L$(TKIO)/lib %c
15:  -L$(SZIP_LIB)
16:
17:LIBES = -ltkselect %
18:  -ltkchdf5algs -ltkchdf5 %c
19:  -ltk -lm -lmfhdf -ldf -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -ltirpc
20:
21:$(MAIN): $(MAIN).o
22:  $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
23:
24:$(MAIN).o: $(MAIN).c
25:  $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
26:
27:clean:
28:  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 5.5.1.

The path to the HDF5 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.

```

$ . /sample_L3_DPR_C "cc" "GPMCOR_DPR_1806_M_D3M_07X.EORC.h5"
lat=2.500000, lon=137.500000
precipRateESurface.mean[0][0][0][63][14]= 2.130813 (mm/hr)
lat=2.500000, lon=137.500000
precipRateESurface.mean[0][1][0][63][14]= 1.616192 (mm/hr)
lat=2.500000, lon=137.500000
precipRateESurface.mean[0][2][0][63][14]= 3.859957 (mm/hr)

```

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

The second argument is the HDF5 file name.

## 5.4 Reading GSMaP\_HDF5 data

### 5.4.1 Source Programs

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

```

1:#include "TKheaders.h"
2:#include "TK_3GSMAPH5.h"
3:int main(int argc, char *argv[])
4:
5: /* declare Data struct */
6: TKINFO granuleHandle3GSMAPH;
7: L3GSMAPH5_GRID L3GSMAPH;
8:
9: /* declare character */
10: char job[256];
11: char inputfile[256];
12:
13: strcpy(job, argv[1]);
14: strcpy(inputfile, argv[2]);
15:
16: /* declare variables */
17: int nlon, nlat;
18: int status;
19: float lat,lon;
20: /* declare array */
21: float hourlyPrecipRateGC[3600][1800];
22:
23: /* HDF file open */
24: status = TKopen(inputfile, "3GSMAPH5", TKREAD, "HDF5", job,
&granuleHandle3GSMAPH, 1);
25: if(status != TK_SUCCESS) {
26: fprintf(stderr, "error:file open error[%s] input3GSMAPH\n", inputfile);
27: }
28:
29: /* Grid data read */
30: status = TKreadGrid( &granuleHandle3GSMAPH, &L3GSMAPH);
31:
32: /* surfacePrecipitation data read */
33: for (nlon=0; nlon<3600; nlon++){
34:   for (nlat=0; nlat<1800; nlat++){
35:     hourlyPrecipRateGC[nlon][nlat] = L3GSMAPH.hourlyPrecipRateGC[nlon][nlat];
36:     if( nlon==1250 && nlat==945 ) {
37:       lat = (180.0/1800.0) * nlat - 90.0 + (180.0/1800.0/2);
38:       lon = (360.0/3600.0) * nlon - 180.0 + (360.0/3600.0/2);
39:       printf("lat=%f, lon=%f\n",lat,lon);
40:       printf("hourlyPrecipRateGC[%d][%d]= %f (mm/hr)\n",
41: nlon, nlat, hourlyPrecipRateGC[nlon][nlat]);
42:     }
43:   }
44: }
45: /* HDF File close*/
46: status = TKclose(&granuleHandle3GSMAPH);
47: return 0;
48: }

```

Include the header file with the corresponding algorithm ID (see Table 5.1). GSMaP is "3GSMAPH5", so include "TK\_3GSMAPH5.h".

hourlyPrecipRateGC read

### 5.4.2 Compilation Method

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

```

1:#cc=icc
2:#CFLAGS=-g -shared-intel -mcmmodel=medium
3:
4:cc=gcc
5:CFLAGS=-g -mcmmodel=medium -fpic -Wall
6:
7:MAIN=. /sample_GSMaP_HDF5_C
8:
9:INC = -I$(HDF5_INC) %
10:  -I$(TKIO)/inc/ccode %c
11:  -I$(xml2) %1}
12:  -I$(SZIP_INC)
13:
14:LIB = -L$(HDF5_LIB) %1}{2}{3}
15:  -L$(TKIO)/lib %c
16:  -L/usr/lib64 %
17:  -L$(SZIP_LIB)
18:
19:LIBES = -ltkselect %
20:  -ltkchdf5algs -ltkchdf5 %c
21:  -ltkc -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2 -ltirpc
22:
23:$(MAIN): $(MAIN).o
24:  $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
25:
26:$(MAIN).o: $(MAIN).c
27:  $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
28:
29:clean:
30:  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 5.8.1.

The path to the HDF5 include and library directories.

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

### 5.4.3 Execution results

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

```

$ . /sample_GSMaP_HDF5_C "hh" "GPMMRG_MAP_2112010000_H_L3S_MCH_05A.h5"
lat=4.55000, lon=-54.950001
hourlyPrecipRateGC[1250][945]= 0.626497 (mm/hr)$

```

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

The second argument is the HDF5 file name.

## 5.5 About the version of PPS Toolkit (TKIO)

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

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

```
DataFormatVersion=bk
```

```
TKCodeBuildVersion=2
```

the version is bk2.

The program changes are in the following three areas

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

```

1: #include "TKheaders.h"
2: #include "TK_2ADPR.h"
3:
4: int main(int argc, char *argv[]){
5:
6: /* declare Data struct */
7: TKINFO granuleHandle2ADPR;
8: L2ADPR_SWATHS L2ADPR;
9:
10: /* declare character */
11: char job[256];
12: char inputfile[256];
13:
14: strcpy(job, argv[1]);
15: strcpy(inputfile, argv[2]);
16:
17: /* declare variables */
18: int year, month, date, dayOfMon, hour, min;
19: int i, j, k;
20: int numOfScan;
21: 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".

```

22:
23: /* declare array */
24: float lat[49], lon[49], precipESurf[49];
25: float zFactorCor[49][176];
26:
27: /* HDF file open */
28: status = TKopen(inputfile, "2ADPR", TKREAD, "HDF5", job, &granuleHandle2ADPR, 1);
29: if(status != TK_SUCCESS) {
30: fprintf(stderr, "error:file open error[%s] input2ADPR¥n", inputfile);
31: }
32:
33: /* META data read */
34: status = TKgetMetaInt(&granuleHandle2ADPR, "NS_SwathHeader",
35: "NumberScansGranule", &numOfScan);
36: if(status != TK_SUCCESS) {
37: fprintf(stderr, "error:file read error[%s]¥n", inputfile);
38: }
39:
40: /* Scan data read */
41: for (i=0; i<numOfScan; i++){
42: status = TKreadScan( &granuleHandle2ADPR, &L2ADPR);
43:
44: /* Scantime Read */
45: year = L2ADPR.NS.ScanTime.Year;
46: month = L2ADPR.NS.ScanTime.Month;
47: dayOfMon = L2ADPR.NS.ScanTime.DayOfMonth;
48: hour = L2ADPR.NS.ScanTime;
49: min = L2ADPR.NS.ScanTime.Minute;
50:
51: for (j=0; j<49; j++){
52:     /* Latitude and Longitude Read */
53: lat[j] = L2ADPR.NS.Latitude[j];
54: lon[j] = L2ADPR.NS.Longitude[j];
55:
56:     /* precipRateESurface,zFactorCorrected Read */
57: precipESurf[j] = L2ADPR.NS.SLV.precipRateESurface[j];
58:     for (k=0; k<176; k++){
59: zFactorCor[j][k] = L2ADPR.NS.SLV.zFactorCorrected[j][k];
60:     }
61:
62:     /* Print the value */
63:     if(i == 3946 && j == 19){
64: printf("Scan=%d, Angle=%d¥n", i, j);
65: printf("lat=%f, lon=%f¥n", lat[j], lon[j]);
66: printf("precipESurf= %f (mm/hr)¥n", precipESurf[j]);
67:     }
68: }
69: }
70:
71: /* HDF File close*/
72: status = TKclose(&granuleHandle2ADPR);
73:
74: return 0;
75:}

```

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.

```

1: #include <stdlib.h>
2: #include <string.h>
3: #include "hdf5.h"
4: int main(){
5:
6:  hid_t file_id, dataset_id;
7:
8:  /* declare character */
9:  char inputfile[64]= "/DPRL2/ GPMCOR_DPR_2112070007_0140_044170_L2S_DD2_07A.h5";

10: char dsetname[32]= "/FS/SLV/precipRateESurface";
11:
12: /* declare variables */
13: int ret;
14:
15: /* declare array */
16: float precipRateESurface[7935][49];
17:
18: /* HDF file open */
19: file_id = H5Fopen(inputfile, H5F_ACC_RDONLY, H5P_DEFAULT);
20:
21: if(file_id < 0 ) {
22:  fprintf(stderr, "error:file open error[%s] %n", inputfile);
23: }
24: 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 HDF5 file to be read.

dsetname sets the name of the data to be read from the HDF5 file.

Defines an area to store read data.

HDF5 file open  
inputfile: HDF5 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.

```

25: {
26: /* dataset open */
27: dataset_id = H5Dopen(file_id, dsetname, H5P_DEFAULT);

28: if(dataset_id < 0 ) {
29: fprintf(stderr, "error:file open error[%s] %n", inputfile);
30: }
31: else
32: {

33: /* precipRateESurface read */
34: ret = H5Dread(dataset_id,H5T_NATIVE_FLOAT,H5S_ALL,H5S_ALL, H5P_DEFAULT,
precipRateESurface);

35: if(ret < 0) {
36: fprintf(stderr, "error:Dataset read error[%s]%n", inputfile);
37: }
38: else
39: {
40: /* file,dataset print */
41: printf("file name= %s %n", inputfile);
42: printf("data set name= %s %n", dsetname);
43: /* precipRateESurface print */
44: printf("precipRateESurface[1491][43]= %f (mm/hr)%n",
precipRateESurface[1491][43]);
45: }

46: /* dataset close */
47: ret = H5Dclose(dataset_id);

48: }
49: /* HDF file close */
50: ret = H5Fclose(file_id);
51: }
52: return 0;
53: }

```

**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.

**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 HDF5 files**  
file\_id: Specify the value obtained by H5Fopen.

### 6.1.2 コンパイル方法

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

```

1:#cc=icc
2:#CFLAGS=-g -shared-intel -mmodel=medium
3:
4:cc=gcc
5:CFLAGS=-g -mmodel=medium -fpic -Wall
6:
7:MAIN=. /sample_HDF5_L2_DPR_C
8:
9:INC = -I$(HDF5_INC) ¥
10:  -I$(SZIP_INC)
11:
12:LIB = -L$(HDF5_LIB) ¥1}{2}{3}{4}{5}{5}
13:  -L$(SZIP_LIB)
14:
15:LIBES = -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2
16:
17:$(MAIN): $(MAIN).o
18:  $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
19:
20:$(MAIN).o: $(MAIN).c
21:  $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
22:
23:clean:
24:  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 HDF5 include directory

The path to the HDF5 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= /DPRL2/ GPMCOR_DPR_2112070007_0140_044170_L2S_DD2_07A.h5
data set name= /FS/SLV/precipRateESurface
precipRateESurface[1491][43]= 3.003061 (mm/hr)

```



## 6.2 Loading L3DPR data

### 6.2.1 Source Programs

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

```

1:#include <stdlib.h>
2:#include <string.h>
3:#include "hdf5.h"
4:int main(){
5:
6: hid_t file_id, dataset_id;
7:
8: /* declare character */
9: char inputfile[64]= "/DPRL3/GPMCOR_DPR_1806_M_D3M_07X.EORC.h5";

10: char dsetname[64]= "/FS/G1/precipRateESurface/mean";
11:
12: /* declare variables */
13: int ret;
14: float lat,lon;
15: /* declare array */
16: float precipRateESurface[3][3][3][72][28];
17:
18: /* HDF file open */
19: file_id = H5Fopen(inputfile, H5F_ACC_RDONLY, H5P_DEFAULT);

20: if(file_id < 0 ) {
21: fprintf(stderr, "error:file open error[%s] \n", inputfile);
22: }
23: else{

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

26: if(file_id < 0 ) {
27: fprintf(stderr, "error:file open error[%s] \n", inputfile);
28: }

```

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 HDF5 file to be read.

dsetname sets the name of the data to be read from the HDF5 file.

HDF5 file open  
inputfile: HDF5 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.

```

29:  else{
30:  /* precipRateESurface read */
31:  ret = H5Dread(dataset_id,H5T_NATIVE_FLOAT,H5S_ALL,H5S_ALL, H5P_DEFAULT,
precipRateESurface);

32:  if(ret < 0) {
33:  fprintf(stderr, "error:Dataset read error[%s]¥n", inputfile);
34:  }
35:  else{
36:  /* file,dataset print */
37:  printf("File name= %s ¥n", inputfile);
38:  printf("Dataset name= %s ¥n", dsetname);
39:
40:  /* lat/lon print */
41:  lat = (140.0/28.0) * 14 - 70.0 + (140.0/28.0/2);
42:  lon = (360.0/72.0) * 63 - 180.0 + (360.0/72.0/2);
43:  printf("lat=%f, lon=%f¥n",lat,lon);
44:
45:  /* precipRateESurface print */
46:  printf("precipRateESurface.mean[0][0][0][63][14]= %f ¥n",
precipRateESurface[0][0][63][14]);
47:  printf("precipRateESurface.mean[0][1][0][63][14]= %f ¥n",
precipRateESurface[0][1][0][63][14]);
48:  }

49:  /* dataset close */
50:  ret = H5Dclose(dataset_id);
51:  }

52:  /* HDF file close */
53:  ret = H5Fclose(file_id);
54:  }
55:  return 0;
56: }

```

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 HDF5 files  
file\_id: Specify the value obtained by H5Fopen.

### 6.2.2 Compilation Method

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

```

1:#cc=icc
2:#CFLAGS=-g -shared-intel -mmodel=medium
3:
4:cc=gcc
5:CFLAGS=-g -mmodel=medium -fpic -Wall
6:
7:MAIN=. /sample_HDF5_L2_DPR_C
8:
9:INC = -I$(HDF5_INC) ¥
10:   -I$(SZIP_INC)
11:
12:LIB = -L$(HDF5_LIB) ¥1}{2}{3}{4}{5}{5
13:   -L$(SZIP_LIB)
14:
15:LIBES = -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2
16:
17:$(MAIN): $(MAIN).o
18:  $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
19:
20:$(MAIN).o: $(MAIN).c
21:  $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
22:
23:clean:
24:  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 HDF5 include directory

The path to the HDF5 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= /DPRL3/GPMCOR_DPR_1806_M_D3M_07X.EORC.h5
data set name= /FS/G1/precipRateESurface/mean
lat=2.500000, lon=137.500000
precipRateESurface.mean[0][0][0][63][14]= 2.130813
precipRateESurface.mean[0][1][0][63][14]= 1.616192$

```

## 6.3 Reading GSMaP\_HDF5 data

### 6.3.1 Source Programs

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

```

1:#include <stdlib.h>
2:#include <string.h>
3:#include "hdf5.h"
4:int main(){
5:
6: hid_t file_id, dataset_id;
7:
8: /* declare character */
9: char inputfile[64]= "/GSMaP/GPMMRG_MAP_2112010000_H_L3S_MCH_05A.h5";

10: char dsetname[64]= "/Grid/hourlyPrecipRateGC";
11:
12: /* declare variables */
13: int ret;
14: float lat,lon;
15: /* declare array */
16: float hourlyPrecipRateGC[3600][1800];
17:

18: /* HDF file open */
19: file_id = H5Fopen(inputfile, H5F_ACC_RDONLY, H5P_DEFAULT);

20: if(file_id < 0 ) {
21: fprintf(stderr, "error:File open error[%s] \n", inputfile);
22: }
23: else{

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

26: if(file_id < 0 ) {
27: fprintf(stderr, "error:Dataset open error[%s] \n", inputfile);
28: }

```

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 HDF5 file to be read.

dsetname sets the name of the data to be read from the HDF5 file.

HDF5 file open  
inputfile: HDF5 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.

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.

```
29: else{
30: /* hourlyPrecipRateGC read */
31: ret = H5Dread(dataset_id,H5T_NATIVE_FLOAT,H5S_ALL,H5S_ALL, H5P_DEFAULT,
hourlyPrecipRateGC);
```

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

```
32: if(ret < 0) {
33: fprintf(stderr, "error:Dataset read error[%s]¥n", inputfile);
34: }
35: else{
36: /* file,dataset print */
37: printf("file name= %s ¥n", inputfile);
38: printf("data set name= %s ¥n", dsetname);
39:
40: lat = (180.0/1800.0) * 945 - 90.0 + (180.0/1800.0/2);
41: lon = (360.0/3600.0) * 1250 - 180.0 + (360.0/3600.0/2);
42: printf("lat=%f, lon=%f¥n",lat,lon);
43:
44: /* hourlyPrecipRateGC print */
45: printf("hourlyPrecipRateGC[1250][945]= %f (mm/hr)¥n",
hourlyPrecipRateGC[1250][945]);
46: }
```

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.

```
47: /* dataset close */
48: ret = H5Dclose(dataset_id);
49: }
```

Close HDF5 files  
 file\_id: Specify the value obtained by H5Fopen.

```
50: /* HDF file close */
51: ret = H5Fclose(file_id);
52: }
53: return 0;
54: }
```

### 6.3.2 Compilation Method

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

```

1:#cc=icc
2:#CFLAGS=-g -shared-intel -mmodel=medium
3:
4:cc=gcc
5:CFLAGS=-g -mmodel=medium -fpic -Wall
6:
7:MAIN=. /sample_HDF5_GSMaP_C
8:
9:INC = -I$(HDF5_INC) ¥
10:  -I$(SZIP_INC)
11:
12:LIB = -L$(HDF5_LIB) ¥1}{2}{3}{4}{5}{5
13:  -L$(SZIP_LIB)
14:
15:LIBES = -lm -lhdf5_hl -lhdf5 -ljpeg -lz -lxml2
16:
17:$(MAIN): $(MAIN).o
18:  $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
19:
20:$(MAIN).o: $(MAIN).c
21:  $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
22:
23:clean:
24:  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 6.3.1.

The path to the HDF5 include directory

The path to the HDF5 library directory

### 6.3.3 Execution Results

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

```

$ . /sample_HDF5_GSMaP_C
file name= /GSMaP/GPMMRG_MAP_2112010000_H_L3S_MCH_05A.h5
data set name= /Grid/hourlyPrecipRateGC
lat=4.550000, lon=-54.950001
hourlyPrecipRateGC[1250][945])= 0.626497 (mm/hr)$

```

## 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 HDF5 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 shell that uses h5dump to create a binary file.

```

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

```

The full path to h5dump is specified.

The directory where the HDF5 files reside is specified.

Specifies the directory to which binary files are output.

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

The extension part of the file name is removed.

Loop for the number of

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

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

```
$ ./dump_L2.sh
HDF5 ". /h5file/GPMCOR_DPR_2112070007_0140_044170_L2S_DD2_07A.h5" {
DATASET "FS/SLV/precipRateESurface" {
  DATATYPE H5T_IEEE_F32LE
  DATASPACE SIMPLE { ( 7935, 49 ) / ( H5S_UNLIMITED, 49 ) }
  DATA {
  }
}
}
```

### 7.1.2 Source Programs

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

```
1:#include <stdio.h>
2:#include <string.h>
3:
4:int main(){
5:
6: /* declare variables */
7: float precipRateESurface[7935][49];
8: FILE *fp;
9:
10: /* binary file declare */
11: char inputfile[128] = ". /binfile/
GPMCOR_DPR_2112070007_0140_044170_L2S_DD2_07A.h5.precipRateESurface";
12:
13: /* binary file open */
14: fp = fopen(inputfile, "rb");
15: if(fp == NULL ) {
16: printf("error:file open error[%s] \n", inputfile);
17: }
18: else
19: {
20: /* file read */
21: fread(precipRateESurface, sizeof(precipRateESurface), 1, fp);
22:
23: /* precipRateESurface print */
24: printf("filename=%s\n",inputfile);
25: printf("precipRateESurface[1491][43]= %f (mm/hr)\n",
precipRateESurface[1491][43]);
26: printf("precipRateESurface[1491][44]= %f (mm/hr)\n",
precipRateESurface[1491][44]);
27:
28: /* binary file close */
29: fclose(fp);
30: }
31: return 0;
32:}
```

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.

```

1:#cc=icc
2:#CFLAGS=-g -shared-intel -mcmmodel=medium
3:
4:cc=gcc
5:CFLAGS=-g -mcmmodel=medium -fpic -Wall
6:
7:MAIN=. /sample_h5dump_L2_C
8:
9:INC = -I$(HDF5_INC) ¥
10:  -I$(SZIP_INC)
11:
12:LIB = -L$(HDF5_LIB) ¥1}{2}{3}{4}{5}{5
13:  -L$(SZIP_LIB)
14:
15:LIBES =-lm      -ljpeg -lz -lxml2
16:
17:$(MAIN): $(MAIN).o
18:  $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
19:
20:$(MAIN).o: $(MAIN).c
21:  $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
22:
23:clean:
24:  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 HDF5 include directory

The path to the HDF5 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=...
/binfile/GPMCOR_DPR_2112070007_0140_044170_L2S_DD2_07A.h5.precipRateESurface
precipRateESurface[1491][43]= 3.003061 (mm/hr)
precipRateESurface[1491][44]= 2.324327 (mm/hr)
$

```

## 7.2 L3 data reading

### 7.2.1 Creating Binary Files

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

```

1:#!/bin/tcsh -f
2:
3:set h5dump_bin=/home/tool/hdf5-1.8.9/bin/h5dump
4:set file_dir=/DPR/STD/L3/
5:set OUTPUT=/h5dump_samplecode/L3/binfile/
6:
7:cd ${file_dir}

8:set file_in=(`ls *1806* |sed 's/.h5/ /'`)
9:cd ${OUTPUT}
10:
11:foreach file ($file_in)
12:echo ${file}
13:$h5dump_bin -d /FS/G1/precipRateESurface/mean -b -o
${file}.precipRateESurface_mean ${file_dir}/${file}.h5
14:end

```

The full path to h5dump is specified.

The directory where the HDF5 files reside is specified.

Specifies the directory to which binary files are output.

Specify if you want to narrow down the files to be processed in the directory where HDF5 files exist. \*1806\* means that only files with "1806" in the file name will be targeted.

The extension part of the file name is removed.

Loop for the number of

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

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

```

$ ./dump_L3.sh
GPMCOR_DPR_1806_M_D3M_07X.EORC
HDF5
"/josui_disk1/trmmauto/sakamoto/guide/h5dump_samplecode/L3/h5file//GPMCOR_DPR_180
6_M_D3M_07X.EORC.h5" {
  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.

```

1:#include <stdio.h>
2:#include <string.h>
3:int main(){
4:
5: /* declare variables */
6: float precipRateESurface[3][3][3][72][28];
7: FILE *fp;
8: int i, j, k, m, n, size;
9: float lat,lon;
10: /* binary file name */
11: char inputfile[128]
12: ="/binfile/GPMCOR_DPR_1806_M_D3M_07X.EORC.precipRateESurface_mean";
13: for(i=0; i<3; i++){
14: for(j=0; j<3; j++){
15: for(k=0; k<3; k++){
16: for(m=0; m<72; m++){
17: for(n=0; n<28; n++){
18: precipRateESurface[i][j][k][m][n]=0;
19: }
20: }
21: }
22: }
23: }
24
25: /* binary file open */
26: fp = fopen(inputfile, "rb");
27: if(fp == NULL ) {
28: printf("error:file open error[%s] %n", inputfile);
29: }
30: else
31: {
32: /* file read */
33: size = fread(precipRateESurface, sizeof(precipRateESurface), 1, fp);
34:
35: /* precipRateESurface print */
36: printf("filename=%s\n",inputfile);
37: lat = (140.0/28.0) * 14 - 70.0 + (140.0/28.0/2);
38: lon = (360.0/72.0) * 63 - 180.0 + (360.0/72.0/2);
39: printf("lat=%f, lon=%f\n",lat,lon);
40: printf("precipRateESurface.mean[0][0][0][63][14]= %f %n",
precipRateESurface[0][0][0][63][14]);
41: printf("precipRateESurface.mean[0][1][0][63][14]= %f %n",
precipRateESurface[0][1][0][63][14]);
42:
43: /* binary file close */
44: fclose(fp);
45: }
46: return 0;
47:}

```

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.

```

1:#cc=icc
2:#CFLAGS=-g -shared-intel -mmodel=medium
3:
4:cc=gcc
5:CFLAGS=-g -mmodel=medium -fpic -Wall
6:
7:MAIN=. /sample_h5dump_L3_C
8:
9:INC = -I$(HDF5_INC) ¥
10: -I$(SZIP_INC)
11:LIB = -L$(HDF5_LIB) ¥1}LIB = -L$(HDF5_LIB)
12: -L$(SZIP_LIB)
13:
14:LIBES =-lm -ljpeg -lz -lxml2
15:
16:$(MAIN): $(MAIN).o
17: $(cc) $(CFLAGS) -o $(MAIN) $(MAIN).o $(INC) $(LIB) $(LIBES)
18:
19:$(MAIN).o: $(MAIN).c
20: $(cc) $(CFLAGS) $(INC) -c $(MAIN).c $(INC) $(LIB) $(LIBES)
21:
22:clean:
23: 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 HDF5 include directory

The path to the HDF5 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=/binfile/GPMCOR_DPR_1806_M_D3M_07X.EORC.precipRateESurface_mean
lat=2.500000, lon=137.500000
precipRateESurface.mean[0][0][0][63][14]= 2.130813
precipRateESurface.mean[0][1][0][63][14]= 1.616192$

```

## revision history

version number	Date	Revised contents	remarks
1	2016/1/26		
2	2017/9/13	<p>1. Introduction: python description added to Table 1.1, flowchart revised accordingly. Table 1.2 Sample code operation check table was added.</p> <p>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</p> <p>4.2.4 Edit configuration file: Change "tkio-3.70.7" to "tkio-x.xx.x".</p>	
3	Mar 15, 2018	3. Related documents and sample programs available: Table 3.1 sample program list added.	
4	1/25/2019	<p>1.-3. Correction due to addition of TRMM and renewal of GPM site</p> <p>4. Table 4.2 Product Version and TKIO Compatible Versions Corrected to list the product version and TKIO compatible version for each product.</p> <p>5. added a list of algorithm IDs and changed the sample programs to include one per level</p>	
5	12/24/2021	<p>1. modified to GSMaP product version 5 and GPM/TRMM product version 7.</p> <p>3. revised availability of related documentation and sample programs</p> <p>Correction of URL for TKIO download</p> <p>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.</p> <p>5. modified V7 changes in the program</p>	
6	1/21/2022	5.-7. Corrections due to modification of sample programs (review of display position, addition of lat/lon to display items, etc.)	