AMSR Data Input Toolkit (ADIT)

User's Guide

Version 3.00

H Edition: Nov. 16, 2012

Japan Aerospace Exploration Agency (JAXA)

MITSUBISHI SPACE SOFTWARE CO., LTD.

Change Record Page

Document Title: AMSR Data Input Toolkit (ADIT) User's Guide Document Date: February 5, 2003			
Issue	Date	Page Affected	Description
Original	26/6/2002	All	Baseline
A	5/2/2003	p1-1	LINUX was added as tested environment in Table 1.2-1.
		p2-3	Makefile for LINUX was added. This version supports C program on LINUX OS .
		〃	Function for correction of leap second between TAI and UTC was added. This version corrects its leap second when SCAN_TIME is converted to UNIX System Time(UTC). Consequently, a leap second information file was added in the directory after expansion of ADIT.
		p2-4	Environment setting procedure on leap second correction was added.
		p3-3	Example of C program for using LINUX OS was added.
		p4-18	Data type of SPC_temp_calc and SPS_temp_calc were changed 4byte real into 8byte real.
		p4-24	Quality flag was revised completely.
		p4-29	CoRegistrationParameterA1 and A2 was added in Table 4.4.1-1.
B	28/5/2003	P5-1～4	Sample programs were added.
C	24/2/2004	p4-24,25	Quality Data in Level2 product is revised.
		p4-30	Added following data in Table 4.4.1-1. ・ CalibrationMethod ・ HTSCorrectionParameterVersion ・ SpillOverParameterVersion ・ MoonLightEffectParameterVersion
D	1/5/2005	p2-4	Installation method of ADIT supported Version 1.07
		p3-1,2 p4-1,3,4,6,7	The description about a 89GHz low frequency equivalent data calculation function was added.
		P4-15	Modify following data in Table 4.3.4-1 ・ pos_orbit
		P5-4	Sample programs were added. Sample data were modified.
E	5/3/2008	P4-33,34	Amend because of misdiscription.
F	2/3/2010	P1-1	HP and DEC were deleted in Table 1.2-1
		P2-3～4	Installation method of ADIT supported Version 2.02.
		P3-4, P3-9	HP and DEC were deleted.
		P3-13～17	The Linux version is added.

G	26/9/2011	P3-1～P3-4, P3-8, P3-14, P4-1～P4-10, P5-3～P5-4	Added functions to read a number of scans.
H	16/11/2012	p1-1	Update compiler version in Table 1.2-1.
		P2-3～4	Installation method of ADIT supported Version 3.00.

Index

1 HDF library and ADIT

1.1 What is HDF?

The AMSR/AMSR-E product is constructed as a Hierarchical Data Format (HDF) file, which was developed by the National Center for Supercomputing Applications (NCSA). If you want to read the data from an HDF file with your C program or Fortran program, you should install the HDF library on your computer. The HDF library is distributed from NCSA with its source codes and/or binary code free of charge. The details for obtaining and installing the HDF library are written in Chapter 2.

1.2 What is ADIT?

There are two ways to read AMSR/AMSR-E data constructed as an HDF file. One is to use only the HDF library, and the other is to use the AMSR Data Input Toolkit (ADIT) which uses the HDF library as internal routine, for reading AMSR/AMSR-E data in your own C program or Fortran program.

ADIT provides functions for reading and storing AMSR/AMSR-E data into local structured variables of one-scan size.

Some data will be converted with a scale factor and saved into the HDF file. ADIT will recognize the scale factor for its conversion and calculate its original value. If you use ADIT to handle AMSR/AMSR-E data in your program codes, you can easily get the correct data. ADIT functionalitys were tested on some of the most popular machines and operating systems, as shown in Table 1.2-1.

Table 1.2-1 ADIT Tested Environments

Platform	OS version	C compiler	FORTRAN compiler	HDF version
Sun	Solaris 10	Oracle Solaris Studio 12.2 cc	Oracle Solaris Studio 12.2 f77	4.2r5
LINUX	2.6.18-194.el5	gcc-4.1.2	GNU:gfortran 4.0.1 PGI:pgf90 6.1-9 Intel:ifort 11.1	4.2r5

2 Installation of HDF library and ADIT

In this Chapter, we describe how to install the HDF library. AMSR/AMSR-E products are produced as HDF files with HDF version 4.2r5 at the Earth Observation Research Center (EORC), JAXA. You should apply the same version of HDF library, but not necessarily for the same revision number. We will show how to install HDF version 4.2r5 in the following section.

2.1 Installation of HDF library

There are two ways to install the HDF library on your machine. One is to install the binary code, the other is to obtain the source code and compiling it for installation. You can obtain both from the HDF library ftp site at NCSA. You may select the suitable type for your machine's OS or source code type.

2.1.1 Installing HDF library from compiled binary

First, please obtain HDF library the binary code for your computer by ftp from the hdfgroup site. You don not need a super user account on your Unix machine to install the HDF library.

If you use a Web browser such as Netscape or Internet Explorer, please set the following URL in its address field to download the files.

```
ftp:// ftp.hdfgroup.org/HDF/HDF_Current/bin
```

Please select the suitable HDF library file for your OS and start to download. After downloading HDF library, please type the following commands to uncompress the downloaded file in your selected directory where you want to install the HDF library.

```
% gunzip 4.2r5-irix64-n32.tar.gz
% tar xvf 4.2r5-irix64-n32.tar
```

If "gunzip" and "tar" are completed without error, a new directory 4.2r5-irix64-n32/ will be created. Its structure and contents are shown in the table below.

4.2r5-irix64-n32/	COPYING	Copyright
	README	README file to use HDF library
	bin/	directory of utility of HDF
	include/	directory of included files of HDF library
	lib/	HDF library directory
	man/	directory of manual of utility
	release_notes/	directory of explanations of HDF library

If the uncompress and installation were successfully completed, you can use the HDF library.

2.1.2 Installation of HDF library from source code

You should prepare an ANSI C compiler for compiling the HDF library. If you don't have an ANSI C compiler, you may use gcc compiler which is freeware from GNU.

You can get HDF library source codes from the following URL via ftp. If you use a Web browser such as Netscape or Internet Explorer, please set the following URL in its address field to download the files.

```
ftp:// ftp.hdfgroup.org/HDF/HDF_Current/bin
```

After downloading the HDF library source codes, please type the following commands to uncompress the downloaded file in your selected directory where you want to install the HDF library.

```
% gunzip HDF4.2r5.tar.gz
% tar xvf HDF4.2r5.tar
```

If "gunzip" and "tar" are completed without error, a new directory 4.2r5-irix64-n32/ will be created. Its structure and contents are shown in the table below.

HDF4.2r5/	COPYING	Copyright
	INSTALL	Installation manual
	MAKEVMS.COM*	
	Makefile.in	
	README	description of directory contents, and so on
	Win32.nofortran.zip*	
	Win32.zip*	
	config/	directory of configuration
	configure*	configure file
	configure.in	
	hdf/	source code directory of HDF library
	install-sh*	
	lib/	
	man/	directory of HDF manuals
	mfhdf/	directory of netCDF
	mkinstalldirs	
	move-if-change*	
	release_notes/	directory of release notes

Before you compile the HDF library, you should confirm the configuration of your operational environment such as the installation directory. If you specify "./configure," the default installation directory is created as "/usr/local," and other directories such as "/usr/local/lib" for the library file, "/usr/local/bin" for the utility file, "/usr/local/man" for the manual file and "/usr/local/include" for the include file are also created. As these files are overwritten in the specific directory, please make sure your specified name directory does not already exist.

In the following sample, we assume that you install the HDF library in your own directory "/home/amsr/work/HDF4.2r5." First, please type the following command to configure the HDF library environment.

```
% ./configure -v –prefix=/home/amsr/work/HDF4.2r5
```

In this step, your HDF library directory is set to "/home/amsr/work/HDF4.2r5" by using "--prefix" option. The "./configure" command creates the most suitable "makefile" for creating the HDF library. The next step is to compile the HDF library.

Please type "make" in the command line to compile the HDF library.

```
% make
```

To confirm that the compiling process is successfully completed, please type the following command.

```
% make test
```

The compiling result will be output to the standard output device. It will be convenient to output this result to your specific file such as "make.test.out."

```
% make test >& smake.test.out
```

Finally, you can install the HDF library by typing the following command.

```
% make install
```

2.2 Installation of ADIT

You can download ADIT from the EORC Web site.

```
http://sharaku.eorc.jaxa.jp/AMSR/tool/index.html
```

After downloading the file, please type the following commands to uncompress ADIT in your current directory.

```
% gunzip ADITv3.00.tar.gz
% tar xvf ADITv3.00.tar
```

You can then confirm the structure of the "ADITv3.00/" directory as follows.

ADITv3.00/	Makefile.SGI	Makefile for IRIX OS
	Makefile.SunOS	Makefile for SUN OS
	Makefile.PGI	
	Makefile.LINUX	Makefile for LINUX OS
	allmake	shell script for making Makefile according to your environment
	install	installer of ADIT
	include/	directory of included ADIT files
	lib/	default installation directory of ADIT library
	src/	directory of ADIT source code
	etc/	directory of Leap second (TAI-UTC) information file
	sample/	directory of sample programs that use ADIT

Please type the following commands to install ADIT.

```
% cd ADITv3.00/
% ./install
```

When you invoke the installer with "./install," you should key in the specific directory name at step A to D below.

```
% ./install
### Start installing AMSR Data Input Toolkit (Ver.3.00) ###
Input the directory of ADIT. (/home/amsr/work/ADITv3.00) ==>                                              A
Input the directory of included files of the HDF library. ==>/home/amsr/work/HDF4.1r2/include            B
Input the directory of library files of the HDF library. ==>/home/amsr/work/HDF4.1r2/lib                 C
Input the directory storing a library of ADIT. (/home/amsr/work/ADITv3.00./lib) ==>                      D

(compiling message)

### Finished installing ADIT. ###
### Created a library of ADIT. (/home/amsr/work/ADITv3.00/lib/libADIT.a) ###
*** Press Enter ===>
%
```

You have to key in the directory name at A to D steps, according to the following directions.

A configuration of the main directory for ADIT installation
This is the first step of configuration, which decides the main directory of ADIT installation directory. The default is the current directory where you Type the commands "./install." Press the return with no input directory name if you like it.

B Configuration of included file path of HDF
Type the directory name of the included file of HDF.

C Configuration of library file path of HDF
Type the directory name of the library file of HDF.

D Configuration of library file path of ADIT
This is the final step of configuration, which decides the library directory of ADIT. Type the directory where you want to install the library of ADIT. If you press return without inputting any directory name, the installer sets the default directory. The default directory is "current_dir/ADITv3.00/lib."

When all configuration steps are processed, the library file will be created automatically in the configured directory. The library file name is "libADIT.a," which you can confirm with the following command. When you confirm the structure of the directory and its ADIT files, your installation of ADIT is completed successfully.

```
% ls -l < configured directory decided at D step>
```

2.3 Setting environment

This library corrects leap second between TAI and UTC by a *leap second information file*. It is adopted for *SCAN_TIME* referring to 4.3.1. It is necessary to set *leap second information file* name to *LEAP_DATA* ,which is environment variable. Its file name should be written by absolute path. An example is shown bellow.

In case of using csh or tcsh, the environment variable is set as follows. For example, csh user should add the bellow sentence in a *.cshrc* file.

```
setenv LEAP_DATA <directory set at A step/etc/tai-utc.dat>
```

Bash user should add the bellow sentence in a .bashrc file. Bsh and ksh user are the same.

```
export LEAP_DATA=<directory set at A step/etc/tai-utc.dat>
```

In addition, the *leap second information* file usually will be updated in a couple of years. So user needs to keep the latest file by yourself getting it from internet, which URL is shown bellow.

```
ftp://maia.usno.navy.mil/ser7/tai-utc.dat
```

If the file is updated, please replace the old file, <directory set at A/etc/tai-utc.dat>, with new one.

3 Programming with ADIT

3.1 Program description

When you use ADIT for AMSR/AMSR-E data handling, please refer to the following descriptions for your own program code.(However, the function that calculates the data of 89GHz low frequency is excluded.)

A Description of header file

You must write the description of the header file for ADIT. The structures of AMSR/AMSR-E level 1B, level 2 and level3 products and some related parameters are specified in this included file.

B Declaration of structures

You may declare the structure of the name you defined in header file of ADIT. After the declaration, you can call your structures in your program code.

C Opening the HDF file

You can open your related HDF file.

D Reading of Metadata

You can input your favorite Metadata in the HDF file to the variable you declared.

E Reading of scan data

AMSR/AMSR-E data stored in the HDF file shall be handled with each or many scan for observation. The observed data stored in the HDF file was input by routines of ADIT to the structure you declared as observed data for scan.

F Closing the HDF file

You can close HDF file, and finish handling the data.

The function that calculates the data of 89GHz makes the observational data for one scene. Therefore, this program is different the procedure from other programs.

A　Opening the HDF file

You can open your related HDF file.

B　Calculation of 89GHz low frequency equivalent data

Read HDF file and parameter file are input, and 89GHz low frequency equivalent data is calculated.

C　Closing the HDF file

You can close HDF file, and finish handling the data.

3.2 C Programming

We will now describe the C sample program.

3.2.1 Example of C program

A Description of header file

#include <AMSR.h>	include file of ADIT

B Declaration of structures

AMSRL1B_SWATH *swath1b;	Left: Structure defined in AMSR.h Right: Structure defined by user

C Opening of HDF file

file_id=openV(HDF file name);	file_id: file_id to access V data of HDF "HDF file name": HDF file name that you will read
sd_id=openSD(HDF file name);	sd_id: sd_id is sd_id to access SD data of HDF "HDF file name": HDF file name that you will read

D Reading of Meta data

status = getATTRIBUTE_NAME_AMSR(sd_id,"LocalGranuleID",granuleID);	
	Description of reading "LocalGranuleID" from Meta data name "sd_id": sd_id that you get at C step. "LocalGranuleID": Meta data name in HDF file "granuleID": Variable defined by user

E Reading of scan data

Reading data for each scan

for(i=0;i<scanno;i++){
   status = getAMSRL1B_SWATH(sd_id,file_id,swath1b,i);
}	

	"sd_id": sd_id that you get at C step. "file_id": file_id that you get at C step "swath1b": structure name defined by user "i": variable for "for loop"

Reading data for number of scans

status = getAMSRL1B_SWATH_line(sd_id,file_id,swath1b,start_scan,end_scan);	
	"sd_id": sd_id that you get at C step. "file_id": file_id that you get at C step "swath1b": structure name defined by user "start_scan": start scan No.(Beginning0) "end_scan": end scan No. (Beginning0)

F　Closing of HDF file

status = closeV(file_id);	Description of closing V data "file_id": file_id that you get at C step
status = closeSD(sd_id);	Description of closing SD data "sd_id": sd_id that you get at C step.

3.2.2　How to compile

In this section, we will explain how to compile the C program for some platforms.

A　SUN (Solaris 8)

Note: Use compiling option "-DSUN -Xc -lnsl -lm"

```
cc -DSUN -Xc -lnsl -o sample1 sample1.c \
-I/home/amsr/work/ADITv3.00/include -I/home/amsr/work/HDF4.2r5/include\
-L/home/amsr/work/ADITv3.00/lib -L/home/amsr/work/HDF4.2r5/lib\
-lADIT -lmfhdf -ldf [-ljpeg] -lz -lm
```

B　SGI (IRIX6.5)

Note: Use compiling option "-DSGI -xansi -O -s -lm"

```
cc -DSGI -xansi -O -s -o sample1 sample1.c \
-I/home/amsr/work/ADITv3.00/include -I/home/amsr/work/HDF4.2r5/include\
-L/home/amsr/work/ADITv3.00/lib -L/home/amsr/work/HDF4.2r5/lib\
-lADIT -lmfhdf -ldf [-ljpeg] -lz -lm
```

C　LINUX (2.2.13-33)

Note:Use compiling option "-DLINUX -ansi -lm"

```
gcc -DLINUX -ansi -o sample1 sample1.c \
-I/home/amsr/work/ADITv3.00/include -I/home/amsr/work/HDF4.2r5/include\
-L/home/amsr/work/ADITv3.00/lib -L/home/amsr/work/HDF4.2r5/lib\
-lADIT -lmfhdf -ldf -ljpeg -lz -lm
```

3.2.3 Sample program code for C

In this section, we note sample1.c.

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

Description of header file for using ADIT

```
int  main(int argc,char *argv[])
{
    int32 i,scanno;
    int32 file_id,sd_id;

    char scannoC[10],granuleID[40];
    char beginD[20],beginT[20];
    char endD[20],endT[20];

    AMSRL1B_SWATH *swath1b;
    SCAN_TIME *scantime;
    SUN_EARTH *sunearth;
    STATUS_L1B *status1b;
    CAL *cal;
    NAVI *navi;
```

Declaration of structures defined in ADIT

Description of variables you will use in this program

```
/* Argument check */
  if(argc!=2) {
    printf("USAGE : sample1 <AMSR/L1B filename>\n");
    exit(1);
  }
```

Description to open SD data and V data in HDF file

```
/* V&SD HDF open */
  if( (file_id=openV(argv[1]))==FAIL ) {
    fprintf(stderr,"Vdata open error(%s)\n",argv[1]);
    exit(1);
  }

  if( (sd_id=openSD(argv[1])) ==FAIL ) {
```

```
    fprintf(stderr,"SDdata open error(%s)\n",argv[1]);
    exit(1);
  }
```

Description of reading "LocalGranuleID" by Metadata name

```
/* coremeta read by name call */
   if((getATTRIBUTE_NAME_AMSR(sd_id,"LocalGranuleID",granuleID))==FAIL)
   exit(1);
  printf("GRANULE ID(call by NAME) : %s\n",granuleID);
```

Description of reading "LocalGranuleID" by Metadata number

```
/* coremeta read by attr_index call */
  if( (getATTRIBUTE_AMSR(sd_id,3,granuleID))==FAIL) exit(1);
  printf("GRANULE ID(call by INDEX) : %s\n",granuleID);

  if( (getATTRIBUTE_AMSR(sd_id,28,scannoC))==FAIL) exit(1);
  scanno=atoi(scannoC);
  printf("SCANNO : %d\n",scanno);

  if (getATTRIBUTE_AMSR(sd_id,7,beginT)==FAIL) exit(1);
  if (getATTRIBUTE_AMSR(sd_id,8,beginD)==FAIL) exit(1);
  if (getATTRIBUTE_AMSR(sd_id,9,endT)  ==FAIL) exit(1);
  if (getATTRIBUTE_AMSR(sd_id,10,endD) ==FAIL) exit(1);
  printf("OBS. TIME : %s %s - %s %s\n",beginD,beginT,endD,endT);

/* memory allocation */
  swath1b = (AMSRL1B_SWATH *)calloc(1,sizeof(AMSRL1B_SWATH));
  scantime = (SCAN_TIME *)  calloc(1,sizeof(SCAN_TIME));
  sunearth = (SUN_EARTH *)  calloc(1,sizeof(SUN_EARTH));
  status1b = (STATUS_L1B *)  calloc(1,sizeof(STATUS_L1B));
  cal = (CAL *)     calloc(1,sizeof(CAL));
  navi = (NAVI *)     calloc(1,sizeof(NAVI));

/* data read every scan */
  for(i=0;i<scanno;i++)
    {
```

```
    printf("SCAN NO. %04d/%d\n",i+1,scanno);
    if (getAMSRL1B_SWATH(sd_id,file_id,swath1b,i)==FAIL) exit(1);
    if (getSCANTIME_AMSR1(file_id,scantime,i)    ==FAIL) exit(1);
    if (getSUN_EARTH(sd_id,sunearth,i)           ==FAIL) exit(1);
    if (getSTATUS_L1B(sd_id,status1b,i)          ==FAIL) exit(1);
    if (getCALIBRATION(sd_id,cal,i)              ==FAIL) exit(1);
    if (getNAVIGATION(sd_id,navi,i)              ==FAIL) exit(1);
    }

/* V&SD close */
  closeV(file_id);
  closeSD(sd_id);

  free(swath1b);
  free(scantime);
  free(sunearth);
  free(status1b);
  free(cal);
  free(navi);

  return 0;
}
```

Descriptions of reading data and input data to the structure for each scan

Description of closing HDF file

＊Please refer to Chapter 5 about the method of accessing a data in a structure.

3.3 Fortran programming (SunOS version, SGI version)

We explain Fortran programming in this section

3.3.1 Example of Fortran program

A Description of header file

include 'AMSR_f.h'	Description of include file of ADIT for Fortran

B Declaration of Structures

record /AMSRL1B_SWATH/	swath1b	Left: Structure defined in AMSR_f.h Right: Structure defined by user

C Opening of HDF file

file_id=openV(HDF file name)	file_id: file_id to access V data of HDF "HDF file name": HDF file name that you will read
sd_id=openSD(HDF file name)	sd_id: sd_id is sd_id to access SD data of HDF "HDF file name": HDF file name that you will read

D Reading of Meta data

status=getATTRIBUTE_NAME_AMSR(sd_id,'LocalGranuleID',granuleID)	
	sd_id: sd_id is sd_id to access SD data of HDF "HDF file name": HDF file name that you will read

E Reading of scan data

Reading data for each scan

```
 do 10 i=0,scanno-1,1
   status=getAMSRL1B_SWATH(sd_id,file_id,swath1b,i)
10 continue
```

	"sd_id": sd_id that you get at C step. "file_id": file_id that you get at C step "swath1b": structure name defined by user "i": Variable for "do loop"

Reading data for number of scans

```
status = getAMSRL1B_SWATH_line(sd_id,file_id,swath1b,start_scan,end_scan);
```

	"sd_id": sd_id that you get at C step. "file_id": file_id that you get at C step "swath1b": structure name defined by user "start_scan": start scan No.(Beginning0) "end_scan": end scan No. (Beginning0)

F　Closing of HDF file

status = closeV(file_id)	Description of closing V data “file_id”: file_id that you get at C step
status = closeSD(sd_id)	Description of closing SD data “sd_id”: sd_id that you get at C step.

3.3.2 How to compile

We explain how to compile the f77 program at some kinds of platforms. About LINUX OS, FORTRAN does not supported.

A SUN (Solaris 8)

Note: Use compiling option " -DSUN -lnsl -lm."

```
f77 -DSUN -lnsl -o sample1f sample1f.f \
-I/home/amsr/work/ADITv3.00/include -I/home/amsr/work/HDF4.2r5/include\
-L/home/amsr/work/ADITv3.00/lib -L/home/amsr/work/HDF4.2r5/lib\
-lADIT -lmfhdf -ldf -lz [-ljpeg] -lm
```

B SGI (IRIX6.5)

Note: Use compiling option "-DSGI -lm."

```
f77 -DSGI -o sample1f sample1f.f \
-I/home/amsr/work/ADITv3.00/include -I/home/amsr/work/HDF4.2r5/include\
-L/home/amsr/work/ADITv3.00/lib -L/home/amsr/work/HDF4.2r5/lib\
-lADIT -lmfhdf -ldf -lz [-ljpeg] -lm
```

3.3.3 Sample program code for Fortran

We explain sample1f.f in this section.

```
      program main

      include 'AMSR_f.h'
```

Description of header file for Fortran to use ADIT

```
      character*30 fname
      data fname/'A2AMS01092011MD_P01B0000000.00'/
      integer status
      integer i,scanno
      integer file_id,sd_id

      character*10 scannoC
      character*40 granuleID
      character*20 beginD,beginT
      character*20 endD,endT

      record /AMSRL1B_SWATH/ swath1b
      record /SCAN_TIME/ scantime
      record /SUN_EARTH/ sunearth
      record /STATUS_L1B/ status1b
      record /CAL/ cal
      record /NAVI/ navi
```

Declaration of structures defined in ADIT

```
C*    V&SD HDF open
      file_id=openV(fname)
      if(file_id .eq. FAIL) then
        write(6,'(a,a,a)') 'Vdata open error(',fname,')'
        stop
      end if
      sd_id=openSD(fname)
      if(sd_id .eq. FAIL) then
        write(6,'(a,a,a)') 'SDdata open error(',fname,')'
        stop
```

Description of opening SD data and V data of HDF

```
      end if
```

Description of reading "LocalGranuleID" by Metadata name

```
C*   coremeta read by name call
      status=getATTRIBUTE_NAME_AMSR(sd_id,'LocalGranuleID',granuleID)
      if(status .eq. FAIL) stop
      write(6,'(a,a27)') 'GRANULE ID(call by NAME) : ',granuleID
```

Description of reading "LocalGranuleID" by Metadata index number

```
C*   coremeta read by attr_index call
      status=getATTRIBUTE_AMSR(sd_id,3,granuleID)
      if(status .eq. FAIL) stop
      write(6,'(a,a27)') 'GRANULE ID(call by INDEX) : ',granuleID

      status=getATTRIBUTE_AMSR(sd_id,28,scannoC)
      if(status .eq. FAIL) stop
          write(6,*) ichar(scannoC(1:1))
      scanno=(ichar(scannoC(1:1))-48)*1000
      +   +(ichar(scannoC(2:2))-48)*100
      +   +(ichar(scannoC(3:3))-48)*10
      +   +(ichar(scannoC(4:4))-48)
      write(6,'(a,i4)') 'SCANNO : ',scanno

      status=getATTRIBUTE_AMSR(sd_id,7,beginD)
      if(status .eq. FAIL) stop
      status=getATTRIBUTE_AMSR(sd_id,8,beginT)
      if(status .eq. FAIL) stop
      status=getATTRIBUTE_AMSR(sd_id,9,endD)
      if(status .eq. FAIL) stop
      status=getATTRIBUTE_AMSR(sd_id,10,endT)
      if(status .eq. FAIL) stop
      write(6,'(a,a12,a,a10,a,a12,a,a10)')
     *   'OBS. TIME : ',beginD,' ',beginT,
     *    ' - ',endD,' ',endT

C*   data read every scan
```

```
    do 10 i=0,scanno-1,1
      write(6,'(a,i4.4,a,i4.4)') 'SCAN NO. ',i+1,'/',scanno
      status=getAMSRL1B_SWATH(sd_id,file_id,swath1b,i)
      if(status .eq. FAIL) stop
      status=getSCANTIME_AMSR1(file_id,scantime,i)
      if(status .eq. FAIL) stop
      status=getSUN_EARTH(sd_id,sunearth,i)
      if(status .eq. FAIL) stop
      status=getSTATUS_L1B(sd_id,status1b,i)
      if(status .eq. FAIL) stop
      status=getCALIBRATION(sd_id,cal,i)
      if(status .eq. FAIL) stop
      status=getNAVIGATION(sd_id,navi,i)
      if(status .eq. FAIL) stop
   10 continue

C*   V&SD close
      status=closeV(file_id)
      status=closeSD(sd_id)

      stop
      end
```

Descriptions of reading data and input data to the structure for each scan

Description of closing SD data and V data of HDF

＊Please refer to Chapter 5 about the method of accessing a data in a structure.

3.4 Fortran programming(Linux version)

We explain Fortran programming in this section

3.4.1 Example of Fortran program

A Description of header file

include ‘AMSR_Linux_f.h’	Description of include file of ADIT for Fortran

B Declaration of Structures

TYPE (AMSRL1B_SWATH)	swath1b	Left: Structure defined in AMSR_Linux_f.h Right: Structure defined by user

C Opening of HDF file

file_id=openV(HDF file name)	file_id: file_id to access V data of HDF “HDF file name”: HDF file name that you will read
sd_id=openSD(HDF file name)	sd_id: sd_id is sd_id to access SD data of HDF “HDF file name”: HDF file name that you will read

D Reading of Meta data

status=getATTRIBUTE_NAME_AMSR(sd_id,'LocalGranuleID',granuleID)	
	sd_id: sd_id is sd_id to access SD data of HDF “HDF file name”: HDF file name that you will read

E Reading of scan data

Reading data for each scan

```
 do 10 i=0,scanno-1,1
   status=getAMSRL1B_SWATH(sd_id,file_id,swath1b,i)
10 continue
```

	“sd_id”: sd_id that you get at C step. “file_id”: file_id that you get at C step “swath1b”: structure name defined by user “i”: Variable for “do loop”

Reading data for number of scans

```
status = getAMSRL1B_SWATH_line(sd_id,file_id,swath1b,start_scan,end_scan);
```

	“sd_id”: sd_id that you get at C step. “file_id”: file_id that you get at C step “swath1b”: structure name defined by user “start_scan”: start scan No.(Beginning0) “end_scan”: end scan No. (Beginning0)

F　Closing of HDF file

status = closeV(file_id)	Description of closing V data “file_id”: file_id that you get at C step
status = closeSD(sd_id)	Description of closing SD data “sd_id”: sd_id that you get at C step.

3.4.2 How to compile

We explain how to compile the f77 program at some kinds of platforms.

A pgi

Note: Use compiling option " -DLINUX -O -lm."

```
pgf90 -DLINUX -O -o sample1f sample1f.f \
-I/home/amsr/work/ADITv3.00/include -I/home/amsr/work/HDF4.2r5/include\
-L/home/amsr/work/ADITv3.00/lib -L/home/amsr/work/HDF4.2r5/lib\
-lADIT -lmfhdf -ldf -lz [-ljpeg] -lm
```

B gun

Note: Use compiling option "-DLINUX -O -lm."

```
gfortran -DLINUX -O -o sample1f sample1f.f \
-I/home/amsr/work/ADITv3.00/include -I/home/amsr/work/HDF4.2r5/include\
-L/home/amsr/work/ADITv3.00/lib -L/home/amsr/work/HDF4.2r5/lib\
-lADIT -lmfhdf -ldf -lz [-ljpeg] -lm
```

C Intel

Note: Use compiling option " -DLINUX_-O -lm."

```
ifort -DLINUX_-O -o sample1f sample1f.f \
-I/home/amsr/work/ADITv3.00/include -I/home/amsr/work/HDF4. 4.2r5/include\
-L/home/amsr/work/ADITv3.00/lib -L/home/amsr/work/HDF4. 4.2r5/lib\
-lADIT-lmfhdf -ldf -lz [-ljpeg] -lm
```

3.4.3 Sample program code for Fortran

We explain sample1f.f in this section.

```
program main

 include 'AMSR_Linux_f.h'
```

Description of header file for Fortran to use ADIT

```
      character*10 scannoC
      character*40 granuleID
      character*20 beginD,beginT
      character*20 endD,endT
      character    buff*100

      data fname /'../data/P1AME030609207MA_P01B0000000.00.sample'/
      data LOCALGRANULEID /'LocalGranuleID'/

   character*10 scannoC
   character*40 granuleID
   character*20 beginD,beginT
   character*20 endD,endT
      character    buff*100

    TYPE (AMSRL1B_SWATH) swath1b
    TYPE (SCAN_TIME)     scantime
    TYPE (SUN_EARTH)     sunearth
    TYPE (STATUS_L1B)    status1b
    TYPE (CAL)           cal_data
    TYPE (NAVI)          navi_data
```

Declaration of structures defined in ADIT

```
      do 100 i=1,100,1
        if( fname(i:i).eq.' ') go to 101
100   continue
101   fname(i:i)=char(0)

      do 110 i=1,100,1
```

```
      if( LOCALGRANULEID(i:i).eq.' ') go to 111
110   continue
111   LOCALGRANULEID(i:i)=char(0)

C*  V&SD HDF open
    file_id=openV(fname)
    if(file_id .eq. FAIL) then
      write(6,'(a,a,a)') 'Vdata open error(',fname,')'
      stop
    end if
    sd_id=openSD(fname)
    if(sd_id .eq. FAIL) then
      write(6,'(a,a,a)') 'SDdata open error(',fname,')'
      stop
    end if

C*  coremeta read by name call
    status=getATTRIBUTE_NAME_AMSR(sd_id,'LocalGranuleID',granuleID)
    if(status .eq. FAIL) stop
    write(6,'(a,a27)') 'GRANULE ID(call by NAME) : ',granuleID

C*  coremeta read by attr_index call
    status=getATTRIBUTE_AMSR(sd_id,3,granuleID)
    if(status .eq. FAIL) stop
    write(6,'(a,a27)') 'GRANULE ID(call by INDEX) : ',granuleID

    status=getATTRIBUTE_AMSR(sd_id,28,scannoC)
    if(status .eq. FAIL) stop
       write(6,*) ichar(scannoC(1:1))
    scanno=(ichar(scannoC(1:1))-48)*1000
    +  +(ichar(scannoC(2:2))-48)*100
    +  +(ichar(scannoC(3:3))-48)*10
    +  +(ichar(scannoC(4:4))-48)
    write(6,'(a,i4)') 'SCANNO : ',scanno

    status=getATTRIBUTE_AMSR(sd_id,7,beginD)
```

Description of opening SD data and V data of HDF

Description of reading "LocalGranuleID" by Metadata name

Description of reading "LocalGranuleID" by Metadata index

```
      if(status .eq. FAIL) stop
      status=getATTRIBUTE_AMSR(sd_id,8,beginT)
      if(status .eq. FAIL) stop
      status=getATTRIBUTE_AMSR(sd_id,9,endD)
      if(status .eq. FAIL) stop
      status=getATTRIBUTE_AMSR(sd_id,10,endT)
      if(status .eq. FAIL) stop
      write(6,'(a,a12,a,a10,a,a12,a,a10)')
     *  'OBS. TIME : ',beginD,' ',beginT,
     *   ' - ',endD,' ',endT

C*  data read every scan
      do 10 i=0,scanno-1,1
        write(6,'(a,i4.4,a,i4.4)') 'SCAN NO. ',i+1,'/',scanno
        status=getAMSRL1B_SWATH(sd_id,file_id,swath1b,i)
        if(status .eq. FAIL) stop
        status=getSCANTIME_AMSR1(file_id,scantime,i)
        if(status .eq. FAIL) stop
        status=getSUN_EARTH(sd_id,sunearth,i)
        if(status .eq. FAIL) stop
        status=getSTATUS_L1B(sd_id,status1b,i)
        if(status .eq. FAIL) stop
        status=getCALIBRATION(sd_id,cal,i)
        if(status .eq. FAIL) stop
        status=getNAVIGATION(sd_id,navi,i)
        if(status .eq. FAIL) stop
     10 continue

C*  V&SD close
      status=closeV(file_id)
      status=closeSD(sd_id)

      stop
      end
```

Descriptions of reading data and input data to the structure for each scan

Description of closing SD data and V data of HDF

＊Please refer to Chapter 5 about the method of accessing a data in a structure.

4 APPENDIX

4.1 Routines defined in ADIT

Specific routines handle the specific level of AMSR/AMSR-E products. The detailed descriptions of these routines are introduced in a later section.

Table 4.1-1 Routine table in ADIT

Product level	Routine name	Description
L1B L2 L3	openV()	File open and initialize for HDF/Vdata
	closeV()	File close for HDF/Vdata
	openSD()	File open and initialize for HDF/SDdata
	closeSD()	File close for HDF/SD data
	getATTRIBUTE_NAME_AMSR()	Read metadata (by "metadata name") (See Section 4.4.)
	getATTRIBUTE_AMSR()	Read metadata (by "attr index") (See Section 4.4.)
L1B	getAMSRL1B_SWATH()	Read HDF and input data to the structure "AMSRL1B_SWATH" (See Section 4.3.2.)
	getSCANTIME_AMSR1()	Read HDF and input data to the structure "SCAN_TIME" (See Section 4.3.1.)
	getSUN_EARTH()	Read HDF and input data to the structure "SUN_EARTH" (See Section 4.3.3.)
	getSTATUS_L1B()	Read HDF and input data to the structure "STATUS_L1B" (See Section 4.3.4.)
	getCALIBRATION()	Read HDF and input data to the structure "CAL" (See Section 4.3.5.)
	getNAVIGATION()	Read HDF and input data to the structure "NAVI" (See Section 4.3.6.)
	getAMSR_89LOW()	The data of 89GHz low frequency corresponding is calculated from 89GHz A horn and B horn.
	getAMSRL1B_SWATH_line()	Read HDF and input data to the structure "AMSRL1B_SWATH" (See Section 4.3.2.)
	getSCANTIME_AMSR1_line ()	Read HDF and input data to the structure "SCAN_TIME" (See Section 4.3.1.)
	getSUN_EARTH_line ()	Read HDF and input data to the structure "SUN_EARTH" (See Section 4.3.3.)
	getSTATUS_L1B_line ()	Read HDF and input data to the structure "STATUS_L1B" (See Section 4.3.4.)
	getCALIBRATION_line ()	Read HDF and input data to the structure "CAL" (See Section 4.3.5.)
	getNAVIGATION_line ()	Read HDF and input data to the structure "NAVI" (See Section 4.3.6.)
L2	getAMSRL2_SWATH()	Read HDF and input data to the structure "AMSRL2_SWATH" (See Section 4.3.7.)
	getSCANTIME_AMSR2()	Read HDF and input data to the structure "SCAN_TIME" (See Section 4.3.1.)
	getSTATUS_L2()	Read HDF and input data to the structure "STATUS_L2" (See Section 4.3.8.)
	getAMSRL2_SWATH_line ()	Read HDF and input data to the structure "AMSRL2_SWATH" (See Section 4.3.7.)
	getSCANTIME_AMSR2_line ()	Read HDF and input data to the structure "SCAN_TIME" (See Section 4.3.1.)
	getSTATUS_L2_line ()	Read HDF and input data to the structure

Product level	Routine name	Description
		"STATUS_L2" (See Section 4.3.8.)
L3	getAMSRL3_MAP()	Read L3 science data and input it to data, which you prepared (See Section 4.3.9.)
	getDIMSIZE()	Access L3 data and get information of L3 science data sizes (See Section 4.3.9.)

4.2 User routine interface in ADIT

User routine interface in ADIT is shown in Table 4.2-1 for C program, and Table 4.2-2 for f77 program.

Table 4.2-1 Routine interface for C

Note: int32 means 4byte int.

Routine name	Parameter	Parameter Type	Input/ Output	Note
file_id = openV (Filename)				
	file_id	int32	Output	If failed, return value is FAIL (or -1)
	File_name	char *	Input	AMSR/AMSR-E HDF Filename
status = closeV (file_id)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	file_id	int32	Input	HDF/Vdata access file id
sd_id = openSD (Filename)				
	sd_id	int32	Output	If failed, return value is FAIL (or -1)
	File_name	char *	Input	AMSR/AMSR-E Filename
status = closeSD (sd_id)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
status = getATTRIBUTE_NAME_AMSR (sd_id,name,value)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	If failed, return value is FAIL (or -1)
	name	char *	Input	Metadata name (See Section 4.4.)
	value	char *	Output	Metadata values (See Section 4.4.)
status = getATTRIBUTE_AMSR (sd_id,index,value)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	If failed, return value is FAIL (or -1)
	index	int32	Input	Metadata index (See Section 4.4.)
	value	char *	Output	Metadata values (See Section 4.4.)
status = getAMSRL1B_SWATH (sd_id,file_id,amsrl1b_swath,scan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	file_id	int32	Input	HDF/Vdata access file id
	amsrl1b_swath	AMSRL1B_SWATH *	Output	information structure defined in ADIT
	scan	int32	Input	Scan No.(Beginning 0)
status = getSCANTIME_AMSR1 (sd_id,scan_time,scan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	scan_time	SCAN_TIME *	Output	information structure defined in ADIT
	scan	int32	Input	Scan No.(Beginning 0)
status = getSUN_EARTH (sd_id,sun_earth,scan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	sun_earth	SUN_EARTH *	Output	information structure defined in ADIT
	scan	int32	Input	Scan No.(Beginning 0)
status = getSTATUS_L1B (sd_id,status_l1b,scan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	status_l1b	STATUS_L1B *	Output	information structure defined in ADIT
	scan	int32	Input	Scan No.(Beginning 0)

Table 4.2-1 Routine interface for C

Note: int32 means 4byte int.

Routine name	Parameter	Parameter Type	Input/ Output	Note
status=getCALIBRATION (sd_id,cal,scan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	cal	CAL *	Output	information structure defined in ADIT
	scan	int32	Input	Scan No.(Beginning 0)
status = getNAVIGATION (sd_id,navi,scan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	navi	NAVI *	Output	information structure defined in ADIT
	scan	int32	Input	Scan No.(Beginning 0)
status = getAMSR_89LOW (sd_id,pol_id,para_name_A,para_name_B,bt_89low)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	pol_id	int32	Input	Polarization id (V-pol : 0, H-pol:1)
	para_name_A	char *	Input	Parameter file for A horn □1
	para_name_B	char *	Input	Parameter file for B horn □1
	bt_89low	float *	Output	equivalent of 89GHz low frequency data
status = getAMSRL1B_SWATH_line (sd_id,file_id,amsrl1b_swath,start_scan,end_scanscan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	file_id	int32	Input	HDF/Vdata access file id
	amsrl1b_swath	AMSRL1B_SWATH *	Output	information structure defined in ADIT
	start_scan	int32	Input	Start scan No.(Beginning0)
	end_scan	Int32	Input	End scan No. (Beginning0)
status = getSCANTIME_AMSR1_line (sd_id,scan_time,start_scan,end_scanscan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	scan_time	SCAN_TIME *	Output	information structure defined in ADIT
	start_scan	int32	Input	Start scan No.(Beginning0)
	end_scan	Int32	Input	End scan No. (Beginning0)
status = getSUN_EARTH_line (sd_id,sun_earth,start_scan,end_scanscan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	sun_earth	SUN_EARTH *	Output	information structure defined in ADIT
	start_scan	int32	Input	Start scan No.(Beginning0)
	end_scan	Int32	Input	End scan No. (Beginning0)
status = getSTATUS_L1B_line (sd_id,status_l1b,start_scan,end_scanscan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	status_l1b	STATUS_L1B *	Output	information structure defined in ADIT
	start_scan	int32	Input	Start scan No.(Beginning0)
	end_scan	Int32	Input	End scan No. (Beginning0)
status=getCALIBRATION_line (sd_id,cal,start_scan,end_scanscan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	cal	CAL *	Output	information structure defined in ADIT
	start_scan	int32	Input	Start scan No.(Beginning0)
	end_scan	Int32	Input	End scan No. (Beginning0)
status = getNAVIGATION_line (sd_id,navi,start_scan,end_scanscan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id

Table 4.2-1 Routine interface for C

Note: int32 means 4byte int.

Routine name	Parameter	Parameter Type	Input/ Output	Note
	navi	NAVI *	Output	information structure defined in ADIT
	start_scan	int32	Input	Start scan No.(Beginning0)
	end_scan	Int32	Input	End scan No. (Beginning0)
status = getAMSRL2_SWATH (sd_id,file_id,amsrl2_swath,scan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	file_id	int32	Input	HDF/Vdata access file id
	amsrl2_swath	AMSRL2_SWATH *	Output	structure defined in ADIT
	scan	int32	Input	Scan No.(Beginning 0)
status = getSCANTIME_AMSR2 (sd_id,scan_time,scan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	scan_time	SCAN_TIME *	Output	structure defined in ADIT
	scan	int32	Input	Scan No.(Beginning 0)
status = getSTATUS_L2 (sd_id,status_l2,scan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	status_l2	STATUS_L2 *	Output	structure defined in ADIT
	scan	int32	Input	Scan No.(Beginning 0)
status = getAMSRL2_SWATH_line (sd_id,file_id,amsrl2_swath,start_scan,end_scanscan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	file_id	int32	Input	HDF/Vdata access file id
	amsrl2_swath	AMSRL2_SWATH *	Output	structure defined in ADIT
	start_scan	int32	Input	Start scan No.(Beginning0)
	end_ scan	Int32int32	Input	End scan No. (Beginning0)
status = getSCANTIME_AMSR2_line (sd_id,scan_time,start_scan,end_scanscan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	scan_time	SCAN_TIME *	Output	structure defined in ADIT
	start_scan	int32	Input	Start scan No.(Beginning0)
	end_ scan	Int32int32	Input	End scan No. (Beginning0)
status = getSTATUS_L2_line (sd_id,status_l2,start_scan,end_scanscan)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	status_l2	STATUS_L2 *	Output	structure defined in ADIT
	start_scan	int32	Input	Start scan No.(Beginning0)
	end_scan	Int32int32	Input	End scan No. (Beginning0)
status = getAMSRL3_MAP (sd_id, file_id, map_2int,map_float,size)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	file_id	int32	Input	HDF/Vdata access file id
	map_2int	short *	Output	L3 science data buffer, which has type of short
	map_float	float *	Output	L3 science data, which has type of float
	size	int	Input	L3 science data size, which has value of n line x n pixel
status = getDIMSIZE (sd_id,ref_no,SIZE)				
	sd_id	int32	Input	HDF/SD access SD id
	ref_no	int32	Input	HDF/SD access SD reference number
	SIZE	int32 *	Output	L3 science data size, which has value

Table 4.2-1 Routine interface for C

Note: int32 means 4byte int.

Routine name	Parameter	Parameter Type	Input/ Output	Note
				of n line x n pixel

※1 The storage directory of parameter files

AMSR

Parameter file for A horn

(install directory)/MAKE_89_LOW_PAM/A2AMS/A289A.prm

Parameter file for B horn

(install directory)/MAKE_89_LOW_PAM/A2AMS/A289B.prm

AMSR-E

Parameter file for A horn

(install directory)/MAKE_89_LOW_PAM/P1AME/P189A.prm

Parameter file for B horn

(install directory)/MAKE_89_LOW_PAM/P1AME/P189B.prm

Table 4.2-2 Routine interface for f77

Note: integer*2 means 2byte int, and real*4 means 4byte real.

Routine name	Parameter	Parameter Type	Input/ Output	Note
file_id = openV (Filename)				
	file_id	integer	Output	If failed, return value is FAIL (or -1)
	File_name	character	Input	AMSR/AMSR-E HDF Filename
status = closeV (file_id)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	file_id	integer	Input	HDF/Vdata access file id
sd_id = openSD (Filename)				
	sd_id	integer	Output	If failed, return value is FAIL (or -1)
	File_name	character	Input	AMSR/AMSR-E Filename
status = closeSD (sd_id)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
status = getATTRIBUTE_NAME_AMSR (sd_id,name,value)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	If failed, return value is FAIL (or -1)
	name	character	Input	Metadata name (See Section 4.4.)
	value	character	Output	Metadata values (See Section 4.4.)
status = getATTRIBUTE_AMSR (sd_id,index,value)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	If failed, return value is FAIL (or -1)
	index	integer	Input	Metadata index (See Section 4.4.)
	value	character	Output	Metadata values (See Section 4.4.)
status = getAMSRL1B_SWATH (sd_id,file_id,amsrl1b_swath,scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	file_id	integer	Input	HDF/Vdata access file id
	amsrl1b_swath	AMSRL1B_SWATH	Output	information structure defined in ADIT
	scan	integer	Input	Scan No.(Beginning 0)
status = getSCANTIME_AMSR1 (sd_id,scan_time,scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	scan_time	SCAN_TIME	Output	information structure defined in ADIT
	scan	integer	Input	Scan No.(Beginning 0)
status = getSUN_EARTH (sd_id,sun_earth,scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	sun_earth	SUN_EARTH	Output	information structure defined in ADIT
	scan	integer	Input	Scan No.(Beginning 0)
status = getSTATUS_L1B (sd_id,status_l1b,scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	status_l1b	STATUS_L1B	Output	information structure defined in ADIT
	scan	integer	Input	Scan No.(Beginning 0)
status=getCALIBRATION (sd_id,cal,scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	cal	CAL	Output	information structure defined in ADIT

Table 4.2-2 Routine interface for f77

Note: integer*2 means 2byte int, and real*4 means 4byte real.

Routine name	Parameter	Parameter Type	Input/ Output	Note
	scan	integer	Input	Scan No.(Beginning 0)
status = getNAVIGATION (sd_id,navi,scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	navi	NAVI	Output	information structure defined in ADIT
	scan	integer	Input	Scan No.(Beginning 0)
status = getAMSR_89LOW (sd_id,pol_id,para_name_A,para_name_B,bt_89low)				
	status	int32	Output	If failed, return value is FAIL (or -1)
	sd_id	int32	Input	HDF/SD access SD id
	pol_id	int32	Input	Polarization id (V-pol : 0, H-pol:1)
	para_name_A	char *	Input	Parameter file for A horn □2
	para_name_B	char *	Input	Parameter file for B horn □2
	bt_89low	float *	Output	equivalent of 89GHz low frequency data
status = getAMSRL1B_SWATH_line (sd_id,file_id,amsrl1b_swath,start_scan,end_scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	file_id	integer	Input	HDF/Vdata access file id
	amsrl1b_swath	AMSRL1B_SWATH	Output	information structure defined in ADIT
	start_scan	integer	Input	Start scan No.(Beginning0)
	end_scan	integer	Input	End scan No. (Beginning0)
status = getSCANTIME_AMSR1_line (sd_id,scan_time,start_scan,end_scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	scan_time	SCAN_TIME	Output	information structure defined in ADIT
	start_scan	integer	Input	Start scan No.(Beginning0)
	end_scan	integer	Input	End scan No. (Beginning0)
status = getSUN_EARTH_line (sd_id,sun_earth,start_scan,end_scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	sun_earth	SUN_EARTH	Output	information structure defined in ADIT
	start_scan	integer	Input	Start scan No.(Beginning0)
	end_scan	integer	Input	End scan No. (Beginning0)
status = getSTATUS_L1B_line (sd_id,status_l1b,start_scan,end_scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	status_l1b	STATUS_L1B	Output	information structure defined in ADIT
	start_scan	integer	Input	Start scan No.(Beginning0)
	end_scan	integer	Input	End scan No. (Beginning0)
status=getCALIBRATION_line (sd_id,cal,start_scan,end_scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	cal	CAL	Output	information structure defined in ADIT
	start_scan	integer	Input	Start scan No.(Beginning0)
	end_scan	integer	Input	End scan No. (Beginning0)
status = getNAVIGATION_line (sd_id,navi,start_scan,end_scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	navi	NAVI	Output	information structure defined in ADIT
	start_scan	integer	Input	Start scan No.(Beginning0)
	end_scan	integer	Input	End scan No. (Beginning0)

Table 4.2-2 Routine interface for f77

Note: integer*2 means 2byte int, and real*4 means 4byte real.

Routine name	Parameter	Parameter Type	Input/ Output	Note
status = getAMSRL2_SWATH (sd_id,file_id,amsrl2_swath,scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	file_id	integer	Input	HDF/Vdata access file id
	amsrl2_swath	AMSRL2_SWATH	Output	structure defined in ADIT
	scan	integer	Input	Scan No.(Beginning 0)
status = getSCANTIME_AMSR2 (sd_id,scan_time,scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	scan_time	SCAN_TIME	Output	structure defined in ADIT
	scan	integer	Input	Scan No.(Beginning 0)
status = getSTATUS_L2 (sd_id,status_l2,scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	status_l2	STATUS_L2	Output	structure defined in ADIT
	scan	integer	Input	Scan No.(Beginning 0)
status = getAMSRL2_SWATH_line (sd_id,file_id,amsrl2_swath,start_scan,end_scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	file_id	integer	Input	HDF/Vdata access file id
	amsrl2_swath	AMSRL2_SWATH	Output	structure defined in ADIT
	start_scan	integer	Input	Start scan No.(Beginning0)
	end_scan	integer	Input	End scan No. (Beginning0)
status = getSCANTIME_AMSR2_line (sd_id,scan_time,start_scan,end_scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	scan_time	SCAN_TIME	Output	structure defined in ADIT
	start_scan	integer	Input	Start scan No.(Beginning0)
	end_scan	integer	Input	End scan No. (Beginning0)
status = getSTATUS_L2_line (sd_id,status_l2,start_scan,end_scan)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	status_l2	STATUS_L2	Output	structure defined in ADIT
	start_scan	integer	Input	Start scan No.(Beginning0)
	end_scan	integer	Input	End scan No. (Beginning0)
status = getAMSRL3_MAP (sd_id, file_id, map_2int,map_float,size)				
	status	integer	Output	If failed, return value is FAIL (or -1)
	sd_id	integer	Input	HDF/SD access SD id
	file_id	integer	Input	HDF/Vdata access file id
	map_2int	integer*2	Output	L3 science data buffer, which has type of 2byte int
	map_float	real*4	Output	L3 science data, which has type of 4byte real
	size	integer	Input	L3 science data size, which has value of n line x n pixel
status = getDIMSIZE (sd_id,ref_no,SIZE)				
	sd_id	integer	Input	HDF/SD access SD id
	ref_no	integer	Input	HDF/SD access SD reference number
	SIZE	integer	Output	L3 science data size, which has value of n line x n pixel

※2 The storage directory of parameter files

AMSR

Parameter file for A horn

(install directory)/MAKE_89_LOW_PAM/A2AMS/A289A.prm

Parameter file for B horn

(install directory)/MAKE_89_LOW_PAM/A2AMS/A289B.prm

AMSR-E

Parameter file for A horn

(install directory)/MAKE_89_LOW_PAM/P1AME/P189A.prm

Parameter file for B horn

(install directory)/MAKE_89_LOW_PAM/P1AME/P189B.prm

4.3 Structure definition in ADIT

You can read AMSR/AMSR-E data in HDF file using structures defined in ADIT. Using these structures, you can read specific data in the HDF file.

Table 4.3-1 Structure definitions

Product level	Structure name	Description
L1B,L2	SCAN_TIME	Information structure of the observational scanning time
L1B	AMSRL1B_SWATH	Information structure of swath data. The member of this structure is as follows. 1. structure “SCAN_TIME” 2. Brightness Temperature 3. Latitude and Longitude of the observation point
	SUN_EARTH	Information structure of angle data related to observation point, sun, and platform. The member of this structure is as follows. 1. Sun Azimuth 2. Sun Elevation 3. Earth Incidence 4. Earth Azimuth 5. Ocean/Lanf flag
	STATUS_L1B	Information structure related to status of the observation data. The member of this structure is as follows. 1. Orbit number 2. Observation Supplement 3. Data Quality
	CAL	Information structure of calibration data. The member of this structure is as follows. 1. Hot-load Count 2. Cold Sky Mirror Count 3. Antenna Temperature Coefficient 4. RX Offset/Gain Count 5. SPC Temperature Count 6. SPS Temperature Count 7. SPC Temperature 8. SPS Temperature
	NAVI	Information structure of navigation data. The member of this structure is as follows. 1. platform position(X,Y,Z) in inertial coordinate 2. platform velocity(Vx,Vy,Vz) in inertial coordinate 3. platform attitude(roll,pitch,yaw) in platform coordinate
L2	AMSRL2_SWATH	Information structure of swath data. The member of this structure is as follows. 1. structure “SCAN_TIME” 2. Geophysical data 3. Latitude and Longitude of the observation point
	STATUS_L2	Information structure of status of the observation data. The member of this structure is as follows. 1. Orbit number 2. Data Quality

4.3.1 L1B, L2 common structure

Table 4.3.1-1 L1B, L2 common structures

Name of structure	member	type	size	Description
SCAN_TIME	koyomi	8byte real	1	total second beginning 1970/1/1 0:0
	year	2byte int	1	year (UT)
	month	2byte int	1	month (UT)
	day	2byte int	1	day (UT)
	hour	2byte int	1	hour (UT)
	minute	2byte int	1	minute (UT)
	second	2byte int	1	second (UT)

(1) SCAN_TIME

"SCAN_TIME" is the structure of scanning start time of the observation. This scanning start time corresponds to the first point of observation in a scan. The member "koyomi" is the total seconds from 1970.01.01.00.00 (Unix system time). Though original scanning start time in L1B products is the total seconds from 1993.01.01.00 by UT (TAI time), ADIT converts TAI time into Unix system time for scanning start time.

4.3.2 AMSRL1B_SWATH (for L1B)

Table 4.3.2-1 AMSRL1B_SWATH

Name of structure	member	type	size	Description
AMSRL1B_SWATH	scan_time	SCAN_TIME	20	structure SCAN_TIME
	tb_low	4byte real	12 x 196	TB data for lower frequency channels Dimension: n channel x n pixel Variable numbers are defined as follows. AMSR-E does not have frequency 50GHz and 52GHz band, therefore these two band data are set to *zero* in every scan and pixel. 1: 6GHz vertical elements TB data [K] 2: 6GHz horizontal elements TB data [K] 3: 10GHz vertical elements TB data [K] 4: 10GHz horizontal elements TB data [K] 5: 18GHz vertical elements TB data [K] 6: 18GHz horizontal elements TB data [K] 7: 23GHz vertical elements TB data [K] 8: 23GHz horizontal elements TB data [K] 9: 36GHz vertical elements TB data [K] 10: 36GHz horizontal elements TB data [K] 11: 50GHz vertical elements TB data [K] 12: 52GHz vertical elements TB data [K]
	tb_high_A	4byte real	2 x 392	TB data for 89GHz channels (A-scan) Dimension: n channel x n pixel Variable numbers are defined as follows. 1: 89GHz A-horn vertical elements TB data [K]

			2: 89GHz A-horn horizontal elements TB data [K]
tb_high_B	4byte real	2 x 392	TB data for 89GHz channels (B-scan) Dimension: n channel x n pixel Variable numbers are defined as follows. 1: 89GHz B-horn vertical elements TB data [K] 2: 89GHz B-horn horizontal elements TB data [K]
latlon_low	4byte real	6 x 2 x 196	Geolocation of the observation point for each lower channels Dimension: n channel x n geolocation variable x n pixel. Lower channel variable numbers are defined as follows. AMSR-E does not have frequency 50GHz and 52GHz band, therefore the 6th data are set to *–9999.0* in every scan and pixel. 1: 6GHz elements data 2: 10GHz elements data 3: 18GHz elements data 4: 23GHz elements data 5: 36GHz elements data 6: 50GHz elements data Geolocation variable numbers are defined as follows. 1: latitude [deg] 2: longitude [deg]
latlon_low_mean	4byte real	2 x 196	Geolocation of the observation *mean* point for lower channels (simple mean value) Dimension: n geolocation variable x n pixel. Variable numbers are defined as follows. 1: latitude [deg] 2: longitude [deg]
latlon_high_A	4byte real	2 x 392	Geolocation of the observation point for 89GHz channels (A-scan). Dimension: n geo-location variable x n pixel Variable numbers are defined as follows. 1: latitude [deg] 2: longitude [deg]
latlon_high_B	4byte real	2 x 392	Geolocation of the observation point for 89GHz channels (B-scan). Dimension: n geo-location variable x n pixel Variable numbers are defined as follows. 1: latitude [deg] 2: longitude [deg]

(1) scan_time

"scan_time" is the structure whose type is the structure "SCAN_TIME."

(2) tb_low

"tb_low," whose dimensions are 12 x 196, is the brightness temperature (TB data) of the lower frequency channels. The size "12" means the number of lower channel variables. The first element is 6GHz-Vertical data, the second is 6GHz-Horizontal data, the third is 10GHz-Vertical data,…., the eleventh is 50GHz-Vertical data, and the twelfth is

52GHz-Vertical data.
The size “196” is the number of samples for each scan. The unit is [K].

Table 4.3.2-1 Brightness temperature data value table

value of data	meaning of data value
positive	normal data
negative	questionable data
-32768	parity error data
-9999	missing packet data

(3) **tb_high_A**

“tb_high_A,” whose dimensions are 2 x 392, is the data of Brightness Temperature of 89GHz channels of A-scan. “2” indicates the dimension of the polarization on the A-horn. The first element is the 89.0GHz-Vertical-A data, and the second is the 89.0GHz-Horizontal-A data.

“392” indicates the number of samples for each scan. Channel element values have the same meaning as in “tb_low.” (See Table 4.3.2-1.)

(4) **tb_high_B**

“tb_high_B,” whose dimensions are 2 x 392, is the data of Brightness Temperature of 89GHz channels of B-scan. “2” indicates the dimension of the polarization on the B-horn. The first element is the 89.0GHz-Vertical-B data, and the second is the 89.0GHz-Horizontal-B data.

“392” indicates the number of samples for each scan. Channel element values have the same meaning as in “tb_low.” (See Table 4.3.2-1.)

(5) **latlon_low, latlon_low_mean**

“latlon_low” is the latitude and longitude of the observation point in a scan for each lower frequency channels, “latlon_low_mean” has representative value (simple mean) of latitude and longitude for all lower frequency channels. There are 196 points in a scan. “latlon_low” and “latlon_low_mean” are in degrees. The latitude ranges from -90 to 90; positive value is north latitude, and negative value is south latitude. The longitude ranges from -180 to 180. (See Table 4.3.2-2 and Table 4.3.2-3.)

Table 4.3.2-2 Latitude data value table

value of data	meaning of data value
-90 ~ 0	north latitude data
0 ~ 90	south latitude data
-9999	missing packet data

Table 4.3.2-3 Longitude data value table

value of data	meaning of data value
-180 ~ 0	west longitude data
0 ~ 180	east longitude data
-9999	missing packet data

(6) latlon_high_A

"latlon_high_A" is the latitude and longitude of the observation point in a scan for 89GHz A-scan. There are 392 points in a scan. "latlon_high_A" has units of [deg]. The latitude ranges from –90 to 90, positive value is north latitude, and negative value is south latitude. The longitude ranges from –180 to 180. (See Table 4.3.2-2 and Table 4.3.2-3.)

(7) latlon_high_B

"latlon_high_B" is the latitude and longitude of the observation point in a scan for 89GHz Bb-scan. There are 392 points in a scan. "latlon_high_B" has units of [deg]. The latitude ranges from -90 to 90, positive value is north latitude, and negative value is south latitude. The longitude ranges from -180 to 180. (See Table 4.3.2-2 and Table 4.3.2-3.)

4.3.3 SUN_EARTH (for L1B)

Table 4.3.3-1 SUN_EARTH

Name of structure	member	type	size	Description
SUN_EARTH	sun_azimuth	4byte real	196	Sun azimuth angle [deg]
	sun_elev	4byte real	196	Sun elevation angle [deg]
	earth_incid	4byte real	196	Earth incident angle [deg]
	earth_azimuth	4byte real	196	Earth azimuth angle [deg]
	ol_flag	2byte int	7 x 196	Ocean/Land flag

(1) sun_azimuth

"sun_azimuth" is the Sun azimuth angle at an observation point. The definition is shown in Fig 4.7.3-1 and the range is 360 degree. This data is calculated corresponding to the observation points of 6.GHz to 36GHz. This value is calculated for the representative point of the lower frequency channels (e.g. latitudes and longitudes in "latlon_low_mean.")

(2) sun_elev

"sun_elev" is the Sun elevation angle at an observation point. The definition is shown in Fig 4.7.3-1 and the range is -90.0 to 180 degrees. Calculated values less than –90.0 degrees will be set to –32687, calculated values exceeding 180 degrees will be set to 32768. For other errors case, it will be set to -32768. The data calculated corresponding to the observation points of 6GHz to 36GHz. This value is calculated for the representative point of the lower frequency channels (e.g. latitudes and longitudes in "latlon_low_mean.")

(3) earth_incid

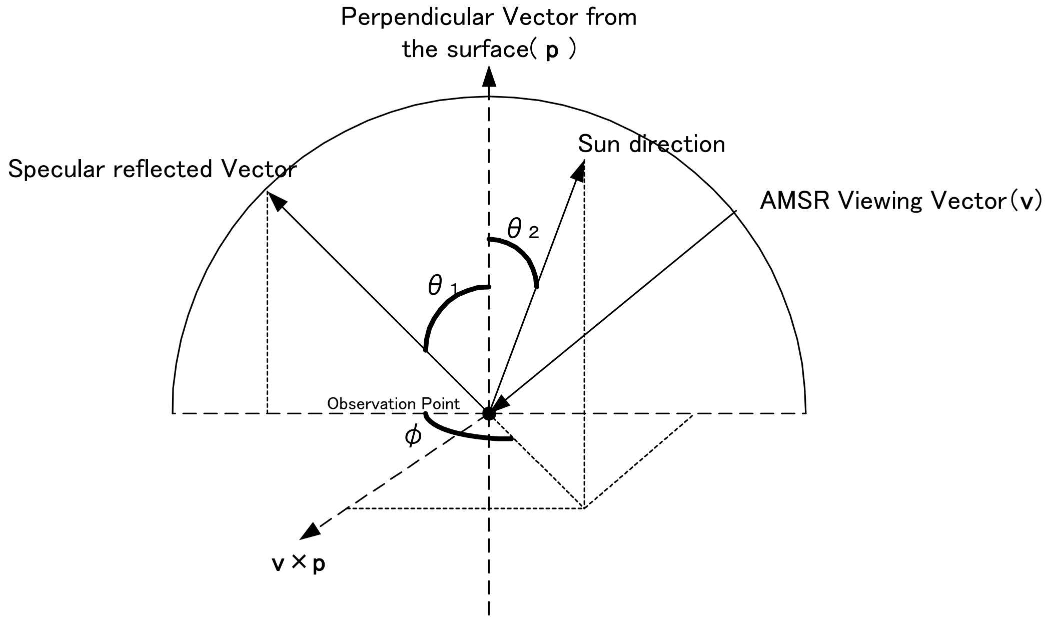
"earth_incid" is the Earth incidence angle at an observation point. The definition is shown in Fig 4.7.3-2 and the range is -90.0 to 180 degrees. Calculated value less than –90.0 will be set to –32687, Calculated values exceeding 180 degrees will be set to 32768. For other errors will be set to -32768. The data calculated corresponding to the observation points of 6GHz to 36GHz. This value is calculated for the representative point of the lower frequency channels (e.g. latitudes and longitudes in "latlon_low_mean.")

(4) earth_azimuth

"earth_azimuth" is the Earth azimuth angle, which is defined as the angle between the north vector and the observation direction vector of AMSR/AMSR-E at an observation point. The definition is shown Fig 4.7.3-2. This data is calculated corresponding to the observation points of 6GHz to 36GHz. This value is calculated for the representative point of the lower frequency channels (e.g. latitudes and longitudes in "latlon_low_mean.")

(4) ol_flag

"ol_flag" is a ratio of land area in the main beam footprint (3dB down beam width) and is expressed on percentage. The data range is from 0 to 100, in abnormal case, data is set to 255. There are 196 stored points in a scan, and these data corresponds to the footprints of 6GHz, 10GHz, 18GHz, 23GHz, 36GHz, 50GHz, and 89GHz-A.



Sun Elevation = $\theta_2 - \theta_1$ (θ_1, θ_2: Absolute Value)

Sun Azimuth = ϕ

(if sun is $\mathbf{v} \times \mathbf{p}$ positive side ; +
negative side ; −)

Fig 4.7.3-1 Definition of Sun Elevation/Azimuth

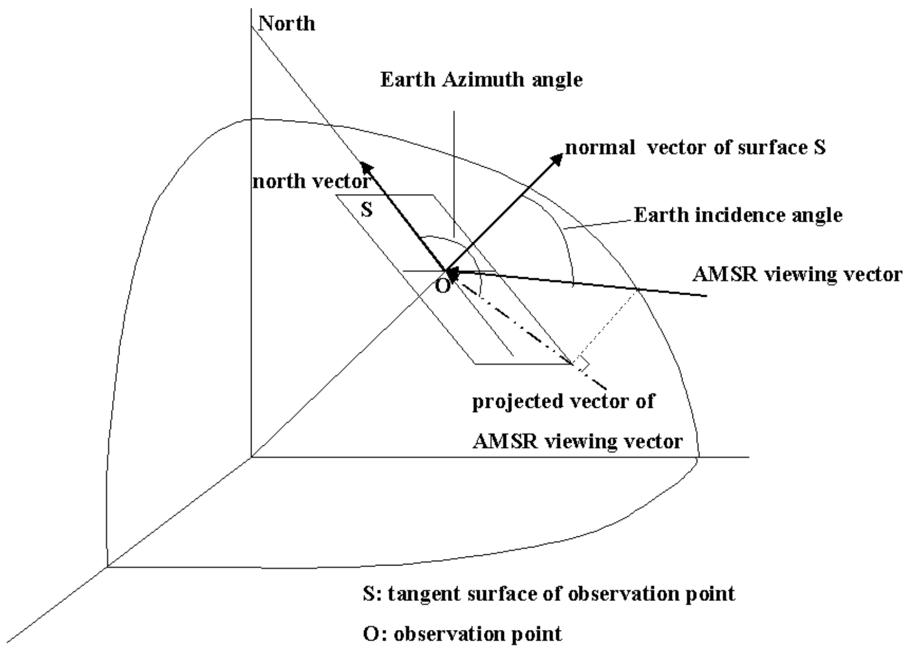


Fig 4.7.3-2 Definition of Earth Azimuth/Incidence

4.3.4 STATUS_L1B (for L1B)

Table 4.3.4-1 STATUS_L1B

Name of structure	member	type	size	Description
STATUS_L1B	pos_orbit	8byte real	1	Orbit No.
	obs_supple	unsigned 2byte int	27	Observation supplement data Dimension: n supplement kind Contents of this array are defined as in Table 4.3.4-2.
	gpsr	2byte int	1	Check value of the GPSR count (0:OK,1:NG) The checking conclusion of GPSR count In the case that the difference of GPSR in before scan and after scan is not satisfied 1.5 ± 1.0sec or -6.5 ± 1.0sec, this flag will be 1.
	hts	2byte int	1	Check value of the HTS count (0:OK,1:NG) The checking conclusion of HTS temperature. In the case that the difference of HTS temperature in before scan and after scan is not satisfied within 0.5°, this flag will be set 1.
	moon_azimuth	4byte real	1	Moon azimuth [deg] The moon direction from Cold Sky Mirror (See Fig 4.3.4-1)
	sun_azimuth	4byte real	1	Sun azimuth [deg] The sun direction from Cold Sky Mirror (See Fig 4.3.4-1)
	tacopulse	4byte real	1	Taco pulse count [count] The average data of Taco pulse count in a product
	quality	4byte real	16 x 4	Statistic values of calibration data Dimension: n channel x n statistic value For AMSR-E, 50GHz vertical and 52GHz vertical element are set to 0.0. Variable numbers are defined as follows. (for n channel) 1: 6GHz vertical elements 2: 6GHz horizontal elements 3: 10GHz vertical elements 4: 10GHz horizontal elements 5: 18GHz vertical elements 6: 18GHz horizontal elements 7: 23GHz vertical elements 8: 23GHz horizontal elements 9: 36GHz vertical elements 10: 36GHz horizontal elements 11: 50GHz vertical elements 12: 52GHz vertical elements 13: 89GHz A-horn vertical elements 14: 89GHz A-horn horizontal elements 15: 89GHz B-horn vertical elements 16: 89GHz B-horn horizontal elements Variable numbers are defined as follows. (for n statistic value)

				1: Cold Sky Mirror Count mean value [count] 2: Hot-load Count mean value [count] 3: Cold Sky Mirror Count root mean square [count] 4: Hot-load Count root mean square [count]

(1) pos_orbit

This data expresses the scanning position in an orbit and is stored in every scan.

Example: The value of "pos_orbit" 100.5 denotes the middle point between orbit number 100. and 101.

(2) obs_supple

"obs_supple" is included in AMSR and AMSR-E telemetry data. This data is stored in every scan. The details of this data are shown in the Table 4.3.4-2.

(3) quality

"quality" is statistic values of calibration data about Cold Sky Mirror Count and Hot-load Count for AMSR and AMSR-E data in every scan. This statistic data contains mean value and root mean square value in unit [count].

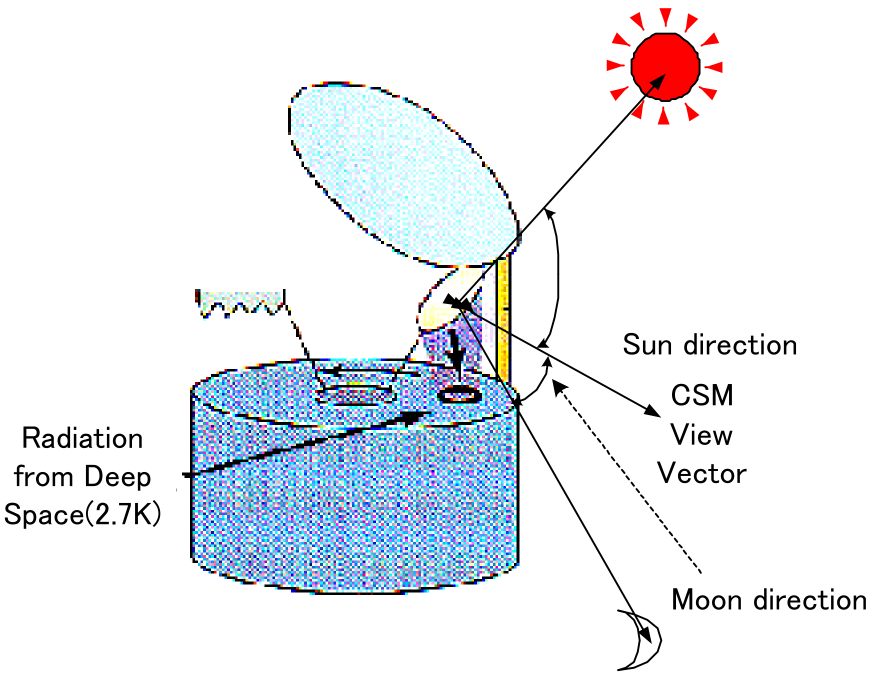


Fig 4.3.4-1 Definition of Sun/Moon direction

Table 4.3.4-2 "obs_supple" data table

Observation supplements array NO.	Description
1	GPSR (Global Positioning System Receiver) count
2	Taco pulse count #1
3	Taco pulse count #2
4	Taco pulse count #3
5	Taco pulse count #4
6	Taco pulse count #5
7	SPC (Signal Processor Control Unit) ON/OFF #1
8	SPC (Signal Processor Control Unit) ON/OFF #2
9	SPC (Signal Processor Control Unit) operation flag
10	SPC (Signal Processor Control Unit) error flag #1
11	SPC (Signal Processor Control Unit) error flag #2
12	SPC (Signal Processor Control Unit) error flag #3
13	SPC (Signal Processor Control Unit) error flag #4
14	Redundancy Switching Control #1
15	Redundancy Switching Control #2
16	SPS(Signal Processor Sensor Unit) ON/OFF #1
17	SPS(Signal Processor Sensor Unit) ON/OFF #2
18	SPS(Signal Processor Sensor Unit) ON/OFF #3
19	SPS(Signal Processor Sensor Unit) ON/OFF #4
20	SPS(Signal Processor Sensor Unit) operation mode
21	RX AGC (Auto Gain Control)/MGC (Manual Gain Control) mode #1
22	RX AGC (Auto Gain Control)/MGC (Manual Gain Control) mode #2
23	SPS(Signal Processor Sensor Unit) operation flag
24	SPS(Signal Processor Sensor Unit) error flag #1
25	SPS(Signal Processor Sensor Unit) error flag #2
26	SPS(Signal Processor Sensor Unit) error flag #3
27	SPS(Signal Processor Sensor Unit) error flag #4

4.3.5 CAL (for L1B)

Table 4.3.5-1 CAL

Name of structure	member	type	size	Description
CAL	ahotload_low	2byte int	12 x 8	Hot-load count for lower frequency channels .(AMSR) Dimension: n channel x n pixel Variable numbers are defined as follows. 1: 6GHz vertical elements data [count] 2: 6GHz horizontal elements data [count] 3: 10GHz vertical elements data [count] 4: 10GHz horizontal elements data [count] 5: 18GHz vertical elements data [count] 6: 18GHz horizontal elements data [count] 7: 23GHz vertical elements data [count] 8: 23GHz horizontal elements data [count] 9: 36GHz vertical elements data [count] 10: 36GHz horizontal elements data [count] 11: 50GHz vertical elements data [count]

			12: 52GHz vertical elements data [count]
ahotload_high_A	2byte int	2 x 16	Hot-load count for 89GHZ channels A-scan (AMSR) Dimension: n channel x n pixel Variable numbers are defined as follows. 1: 89GHz A-horn vertical elements data [count] 2: 89GHz A-horn horizontal elements data [count]
ahotload_high_B	2byte int	2 x 16	Hot-load count for 89GHZ channels B-scan (AMSR) Dimension: n channel x n pixel Variable numbers are defined as follows. 1: 89GHz B-horn vertical elements data [count] 2: 89GHz B-horn horizontal elements data [count]
acoldsky_low	2byte int	12 x 8	Cold sky mirror count for lower frequency channels (AMSR) Dimension: n channel x n pixel Variable numbers are defined as follows. 1: 6GHz vertical elements data [count] 2: 6GHz horizontal elements data [count] 3: 10GHz vertical elements data [count] 4: 10GHz horizontal elements data [count] 5: 18GHz vertical elements data [count] 6: 18GHz horizontal elements data [count] 7: 23GHz vertical elements data [count] 8: 23GHz horizontal elements data [count] 9: 36GHz vertical elements data [count] 10: 36GHz horizontal elements data [count] 11: 50GHz vertical elements data [count] 12: 52GHz vertical elements data [count]
acoldsky_high_A	2byte int	2 x 16	Cold sky mirror count for 89GHZ channels A-scan (AMSR) Dimension: n channel x n pixel Variable numbers are defined as follows. 1: 89GHz A-horn vertical elements data [count] 2: 89GHz A-horn horizontal elements data [count]
acoldsky_high_B	2byte int	2 x 16	Cold sky mirror count for 89GHZ channels B-scan (AMSR) Dimension: n channel x n pixel Variable numbers are defined as follows. 1: 89GHz B-horn vertical elements data [count] 2: 89GHz B-horn horizontal elements data [count]
ehotload_low	2byte int	12 x 16	Hot-load count for lower frequency channels (AMSR-E) Dimension: n channel x n pixel Variable numbers are defined as follows. AMSR-E data does not have 50GHz and 52GHz frequency bands, therefore these two band data are set to *zero* in every scan and pixel. 1: 6GHz vertical elements data [count] 2: 6GHz horizontal elements data [count] 3: 10GHz vertical elements data [count] 4: 10GHz horizontal elements data [count] 5: 18GHz vertical elements data [count] 6: 18GHz horizontal elements data [count] 7: 23GHz vertical elements data [count] 8: 23GHz horizontal elements data [count] 9: 36GHz vertical elements data [count] 10: 36GHz horizontal elements data [count] 11: 50GHz vertical elements data [count] 12: 52GHz vertical elements data [count]

ehotload_high_A	2byte int	2 x 32	Hot-load count for 89GHZ channels A-scan (AMSR-E) Dimension: n channel x n pixel Variable numbers are defined as follows. 1: 89GHz A-horn vertical elements data [count] 2: 89GHz A-horn horizontal elements data [count]
ehotload_high_B	2byte int	2 x 32	Hot-load count for 89GHZ channels B-scan (AMSR-E) Dimension: n channel x n pixel Variable numbers are defined as follows. 1: 89GHz B-horn vertical elements data [count] 2: 89GHz B-horn horizontal elements data [count]
ecoldsky_low	2byte int	12 x 16	Cold sky mirror count for lower frequency channels (AMSR-E) Dimension: n channel x n pixel Variable numbers are defined as follows. AMSR-E does not have 50GHz and 52GHz frequency bands, therefore these two band data are set to *zero* in every scan and pixel. 1: 6GHz vertical elements data [count] 2: 6GHz horizontal elements data [count] 3: 10GHz vertical elements data [count] 4: 10GHz horizontal elements data [count] 5: 18GHz vertical elements data [count] 6: 18GHz horizontal elements data [count] 7: 23GHz vertical elements data [count] 8: 23GHz horizontal elements data [count] 9: 36GHz vertical elements data [count] 10: 36GHz horizontal elements data [count] 11: 50GHz vertical elements data [count] 12: 52GHz vertical elements data [count]
ecoldsky_high_A	2byte int	2 x 32	Cold sky mirror count for 89GHZ channels A-scan (AMSR-E) Dimension: n channel x n pixel Variable numbers are defined as follows. 1: 89GHz A-horn vertical elements data [count] 2: 89GHz A-horn horizontal elements data [count]
ecoldsky_high_B	2byte int	2 x 32	Cold sky mirror count for 89GHZ channels B-scan (AMSR-E) Dimension: n channel x n pixel Variable numbers are defined as follows. 1: 89GHz B-horn vertical elements data [count] 2: 89GHz B-horn horizontal elements data [count]
ant_temp_coef	4byte real	32	Antenna Temperature Coefficient for every channel in a scan Dimension: n channel offset/gain Variables are listed in Table 4.3.5-4. The unit of gain is [K/count] and the unit of offset is [K].
offset_gain	unsigned 2byte int	32	Rx Offset/Gain Count for every channel in a scan Dimension: n channel offset/gain Variables are listed in Table 4.3.5-1. Unit is [count].
SPC_temp_cnt	2byte int	20	Temperature counts of Signal processor control unit array style n kind Variables are listed in Table 4.3.5-5. Unit is [count].
SPS_temp_cnt	2byte int	32	Temperature counts of Signal processor sensor unit

			array style n kind Variable is defined as Table 4.3.5-6. Unit is [count].
SPC_temp_calc	8byte real	20	Temperature of Signal processor control unit calculated from "SPC_temp_cnt." Variables are listed in Table 4.3.5-5. Unit is [ºC].
SPS_temp_calc	8byte real	32	Temperature of Signal processor sensor unit calculated from "SPS_temp_cnt." Variables are listed in Table 4.3.5-6. Unit is [ºC].

(1) a[e]hotload_low, a[e]hotload_high_A[B]

There are 8 (AMSR) or 16 (AMSR-E) points in one scan for lower frequency channels and 16 (AMSR) or 32 (AMSR-E) points in one scan for 89GHz channels. Hot-load count data are observed digital counts of the High Temperature nose Source (HTS, e.g., hot load) in a scan.

If you use AMSR L1B, ADIT uses automatically the members of "acoldsky_low," "ahotload_high_A," and "ahotload_high_B." In using AMSR-E L1B, ADIT uses automatically the members of "ehotload_low," "ehotload_high_A" and "ehotload_high_B". The meaning of each lower frequency channel element's value is shown in Table 4.3.5-2.

Table 4.3.5-2 Hot-load counts data value

value of data	meaning of data value
positive	normal data
negative	questionable data
-32768	parity error data
0	missing packet data

(2) a[e]coldsky_low, a[e]coldsky_high_A[B]

There are 8 (AMSR) or 16 (AMSR-E) points in one scan for lower frequency channels and 16 (AMSR) or 32 (AMSR-E) points in one scan for 89GHz channels. Cold Sky Mirror Count data are observed digital counts of Deep space (Cosmic Microwave Background) using the Clod Sky Mirror in a scan.

If you use AMSR L1B, ADIT uses automatically the members of "acoldsky_low," "acoldsky_high_A" and "acoldsky_high_B." In using AMSR-E L1B, ADIT uses automatically the members of "ecoldsky_low," "ecoldsky_high_A" and "ecoldsky_high_B." The meaning of each lower frequency channel element's value is shown in Table 4.3.5-3.

Table 4.3.5-3 Cold sky mirror counts data value

value of data	meaning of data value
positive	questionable data
negative	normal data
32767	parity error data
0	missing packet data

(3) ant_temp_coef

"ant_temp_coef" is the coefficient for converting from observation counts to antenna temperature. The coefficients are the slope and offset for every frequency and polarization channel, and are stored in every scan. This data array is defined in Table 4.3.5-4.

(4) offset_gain

"offset_gain" is the receiver offset/gain data measured every scan. This data array is defined in Table 4.3.5-4.

(5) SPC_temp_cnt, SPC_temp_calc

"SPC_temp_cnt" is the temperature count data of the signal-processor control unit. The "SPC_temp_calc" is calculated physical temperatures from "SPC_temp_cnt." Contents of this data are listed in Table 4.3.5-5.

(6) SPS_temp_cnt,SPS_temp_calc

"SPS_temp_cnt" is the temperature count data of the signal-processor sensor unit. The "SPS_temp_calc" is calculated value from "SPS_temp_cnt." Contents of this data are listed in Table 4.3.5-6.

Table 4.3.5-4" ant_temp_coef"/" offset_gain" data table

Variable No. of n channel offsetgain	Description
1	6GHz vertical elements of offset
2	6GHz vertical elements of gain [slope]
3	6GHz horizontal elements of offset
4	6GHz horizontal elements of gain [slope]
5	10GHz vertical elements of offset
6	10GHz vertical elements of gain [slope]
7	10GHz horizontal elements of offset
8	10GHz horizontal elements of gain [slope]
9	18GHz vertical elements of offset
10	18GHz vertical elements of gain [slope]
11	18GHz horizontal elements of offset
12	18GHz horizontal elements of gain [slope]
13	23GHz vertical elements of offset
14	23GHz vertical elements of gain [slope]
15	23GHz horizontal elements of offset
16	23GHz horizontal elements of gain [slope]
17	36GHz vertical elements of offset
18	36GHz vertical elements of gain [slope]
19	36GHz horizontal elements of offset
20	36GHz horizontal elements of gain [slope]
21	50GHz vertical elements of offset
22	50GHz vertical elements of gain [slope]
23	52GHz vertical elements of offset

24	52GHz vertical elements of gain [slope]
25	89GHz A-horn vertical elements of offset
26	89GHz A-horn vertical elements of gain [slope]
27	89GHz A-horn horizontal elements of offset
28	89GHz A-horn horizontal elements of gain [slope]
29	89GHz B-horn vertical elements of offset
30	89GHz B-horn vertical elements of gain [slope]
31	89GHz B-horn horizontal elements of offset
32	89GHz B-horn horizontal elements of gain [slope]

Table 4.3.5-5 "SPC_temp_cnt"/"SPC_temp_calc" data table

Variable No. of n kind	Description
1	Thermistor #1 SPC A temperature
2	Thermistor #2 SPC B temperature
3	Thermistor #3 TCC temperature
4	Thermistor #4 PDUC temperature
5	Thermistor #5 ADASTATOR temperature
6	Thermistor #7 MWA Wheel temperature
7	Thermistor #8 MWA Bearing temperature
8	Thermistor #9 ADE temperature
9	Thermistor #11 Control STR temperature
10	Thermistor #12 Control STR temperature
11	Thermistor #13 Control STR temperature
12	Thermistor #14 Control STR temperature
13	Platinum sensor #1 HTS temperature 1
14	Platinum sensor #2 HTS temperature 2
15	Platinum sensor #3 HTS temperature 3
16	Platinum sensor #4 HTS temperature 4
17	Platinum sensor #5 HTS temperature 5
18	Platinum sensor #6 HTS temperature 6
19	Platinum sensor #7 HTS temperature 7
20	Platinum sensor #8 HTS temperature 8

Table 4.3.5-6 "SPS_temp_cnt"/"SPS_temp_calc" data table

n kind variable No.	Description
1	Thermistor #1 SPS temperature
2	Thermistor #2 PUDC temperature
3	Thermistor #3 TCS temperature
4	Thermistor #4 DC/DC RX 1 temperature
5	Thermistor #5 DC/DC RX 2 temperature
6	Thermistor #6 6G LNA temperature
7	Thermistor #7 10G LNA temperature
8	Thermistor #8 50G LNA temperature
9	Thermistor #9 89G H LNA1 temperature
10	Thermistor #10 89G H LNA2 temperature
11	Thermistor #11 89G V LNA1 temperature
12	Thermistor #12 89G V LNA2 temperature
13	Thermistor #13 Sensor STR3 temperature
14	Thermistor #14 Control STR4 temperature
15	Thermistor #15 ADA ROT A temperature
16	Thermistor #16 ADA ROT B temperature
17	Platinum sensor #1 6G RX temperature
18	Platinum sensor #2 10G RX temperature

19	Platinum sensor #3 18G RX temperature
20	Platinum sensor #4 23G RX temperature
21	Platinum sensor #5 36G RX temperature
22	Platinum sensor #6 50G RX temperature
23	Platinum sensor #7 89G RX1 temperature
24	Platinum sensor #8 89G RX2 temperature
25	Platinum sensor #9 MREF 1 temperature
26	Platinum sensor #10 MREF 2 temperature
27	Platinum sensor #11 MREF 3 temperature
28	Platinum sensor #12 MREF 4 temperature
29	Platinum sensor #13 FEED 1 temperature
30	Platinum sensor #14 FEED 2 temperature
31	Platinum sensor #15 sensor STR1 temperature
32	Platinum sensor #16 sensor STR2 temperature

4.3.6 NAVI (for L1B)

"NAVI" is the structure of the navigation data of the platform. The structure is defined in Table 4.3.6-1.

Table 4.3.6-1 NAVI

Name of structure	member	type	size	Description
NAVI	posX	4byte real	1	platform position in X coordinate [m]
	posY	4byte real	1	platform position in Y coordinate [m]
	posZ	4byte real	1	platform position in Z coordinate [m]
	velX	4byte real	1	platform velocity in X coordinate [m/s]
	velY	4byte real	1	platform velocity in Y coordinate [m/s]
	velZ	4byte real	1	platform velocity in Z coordinate [m/s]
	roll	4byte real	1	platform attitude of roll angle [deg]
	pitch	4byte real	1	platform attitude of pitch angle [deg]
	yaw	4byte real	1	platform attitude of yaw angle [deg]

(1) NAVI

The structure "NAVI" contains position, velocity and attitude data of the platform. Position and velocity data are expressed in an inertia co-ordinate system and stored corresponding to the structure "SCAN_TIME." The unit of position ("posX," "posY," "posZ") is [m], and velocity ("velX," "velY,","velZ") is [m/s]. Three kinds of navigation data are used to acquire position data and velocity data, GPS, ELMD and ELMP. Metadata (attribute name is "EphemerisType") specifies which data is stored.

Attitude data ("roll," "pitch," "yaw") have units of [deg]. The value "roll" is the direction of flight, "yaw" is the direction of the nadir, and "pitch" is the direction of "yaw" x "Roll."

Table 4.3.6-2 Navigation data value table

value of data	meaning of data value
except -9999	normal data
-9999	missing packet data

4.3.7 AMSRL2_SWATH (for L2)

Table 4.3.7-1 AMSRL2_SWATH

Name of structure	member	type	size	Description
AMSRL2_SWATH	scan_time	SCAN_TIME	20	Structure of SCAN_TIME
	geophys	4byte real	3 x 196	Geophysical data in a scan. Dimension: n rank x n pixel Variable numbers are defined as follows. 1: geophysical data 2: depends on PI 3: depends on PI
	latlon_low	4byte real	2 x 196	Geolocation of the observation *mean* point for lower channels (simple mean value) Dimension: n geolocation variable x n pixel. Variable numbers are defined as follows. 1: latitude [deg] 2: longitude [deg]

(1) scan_time

"scan_time" is the structure "SCAN_TIME."

(2) geophys

"geophys" is the geophysical data in a scan. There are several kinds of geophysical parameters. (See Table 4.3.7-2.)

Table 4.3.7-2 Geophysical quantity parameters and L2 product code

geophysical parameters	product code	unit	maximum value	minimum value
Water Vapor	WV0	kg/m^2	0	70
Cloud Liquid Water	CLW	kg/m^2	0	1.0
Amount of Precipitation	AP0	mm/h	0	100
Sea Surface Wind	SSW	m/s	0	30
Sea Surface Temperature	SST	ºC	-2	35
Ice Concentration	IC0	%	0	100
Soil Moisture	SM0	g/cm^3	0	To be defined
Snow Water Equivalence	SWE	mm	0	10000

(3) latlon_low

"latlon_low" includes the latitude and longitude of the representative observation point for lower frequency channels in a scan. There are 196 points in a scan. The "latlon_low" has units of [deg]. The latitude ranges from –90 to 90, positive value is north latitude, and negative value is south latitude. The longitude ranges from –180 to 180. (See Table 4.3.2-2 and Table 4.3.2-3.)

4.3.8 STATUS_L2 (for L2)

Table 4.3.8-1 STATUS_L2

Name of struct	member	type	size	Description
STATUS_L2	pos_orbit	4byte real	1	Orbit No.
	quality	1byte int	3 x 196 x 8	Quality flag corresponding to each point of geophysical quantity data. Dimension: n rank x n pixel n bit-position Variable numbers of n rank are defined as follows. 1: geophysical data quality flag 2: depends on PI(*1) 3: depends on PI(*1) Variable numbers of n bit-position are shown in Table 4.3.8-2.

(*1) PI: Proposal Instructor

(1) pos_orbit

This data express the scanning position in an orbit and is stored every scan.

Example: "pos_orbit" 100.5 denotes the middle point between orbit number 100. and 101.

(2) quality

"quality" is the quality flag for L2 data in every scan. (See Table 4.3.8-2.)

Table 4.3.8-2 Quality Flag in detail

Data	Bit position							
	7	6	5	4	3	2	1	0
WV	Land/coast	Abnormal brightness temperature	Sea ice	Abnormal supplementary-sea_surface temperature-wind at sea-temperature of 850hPa	Abnormal calculation of sea_surface emissivity	Cloud	Rainfall	Low precision
CLW	IRETX(2) means no retrieval was done	ISUR2 means land contamination	IICE means sea ice	IOOB(2) means TB OOB	Unused	Unused	Unused	Unused
AP	Tb OK/Bad Tb	no rain/light rain	no rain/heavier rain	retrieval done/no retrieval	Unused	Unused	Unused	Unused
SSW	Land area	Sea ice	Sun glitter	Rain	no data of w6 in correcting wind direction	incident angle error	abnormal wind speed	not used
SST	Land area	Sea ice	Sun glitter	Rain	Wind	Incident angle	Abnormal SST	Not enough number for average TB
IC	No calculation took place	Invalid brightness temperature	Land location	Latitude is out of ice range	Pixel is out of sea area	High SST	Unused	Unused

Data	Bit position							
	7	6	5	4	3	2	1	0
SWE	0:No snow (normal retrieval) 1:Water 2:Snow impossible 3:Permanent ice 4:Surface temperature too warm 5:Heavy forest 6:Mountainous region 7:Rain 8:Wet snow 9:Dry snow (currently unused) 10:Wet soil 11:Dry soil (currently unused) 12:Tb out of range 13:Snow possible 14:Satellite attitude out of range * 15:Missing Tb values *							
SM	Unused							

4.3.9 L3 Science data

There is no structure defined in ADIT for L3. But ADIT provides L3 science data as a 4byte real data, whose size is corresponding to geophysical parameters and map projection type. (See Table 4.3.9-1.)

Table 4.3.9-1 L3 science data size

geophysical parameters	product code	map projection type	size line x pixel	unit
Brightness Temperature	TB	Equirectangular	721 x 1440	K
		Polar stereo in the northern hemisphere	448 x 304	
		Polar stereo in the southern hemisphere	332 x 316	
Water Vapor	WV0	Equirectangular	721 x 1440	kg/m^2
Cloud Liquid Water	CLW	Equirectangular	721 x 1440	kg/m^2
Amount of Precipitation	AP0	Equirectangular	721 x 1440	mm/h
Sea Surface Wind	SSW	Equirectangular	721 x 1440	m/s
Sea Surface Temperature	SST	Equirectangular	721 x 1440	℃
Ice Concentration	IC0	Polar stereo in the northern hemisphere	448 x 304	%
		Polar stereo in the southern hemisphere	332 x 316	
Soil Moisture	SM0	Equirectangular	721 x 1440	g/cm^3
Snow Water Equivalence	SWE	Equirectangular	721 x 1440	cm
		Polar stereo in the southern hemisphere	573 x 431	mm

4.4 Metadata

4.4.1 L1B Metadata

Table 4.4.1-1 L1B Metadata

metadata index	metadata name	Description	metadata values (example)
0	ShortName	product name	AMSR-L1B
1	VersionID	product version ID	RELEASE1
2	SizeMBECSDataGranule	product size (MB)	29.6
3	LocalGranuleID	Local Granule ID	A2AMS01091857MD_P01B0000000
4	ProcessingLevelID	Processing Level ID	L1B
5	ReprocessingActual	Reprocessing Actual (UTC)	1998-02-20
6	ProductionDateTime	Production Date Time (UTC)	1998-02-04-T00:00:00.00Z
7	RangeBeginningTime	Range Beginning Time (UTC)	00:00:00.00Z
8	RangeBeginningDate	Range Beginning Date (UTC)	1998-02-04
9	RangeEndingTime	Range Ending Time (UTC)	01:00:00.00Z
10	RangeEndingDate	Range Ending Date (UTC)	1998-02-04
11	GringPointLatitude	Gring Point of Latitude	-90.0,-90.0,-90.0,-90.0,-90.0,-90.0,-90.0,-90.0
12	GringPointLongitude	Gring Point of Longitude	-180.0,-180.0,-180.0,-180.0,-180.0,-180.0,-180.0,-180.0
13	PGEName	Name of L1B Process Software	L1B_Process_Software
14	PGEVersion	Version of L1B Process Software	3222222***11111*11
15	InputPointer	Inputted file name	A2AMS01091857MD_P01A0000000
16	ProcessingCenter	Data Processing Center	NASDA EOC
17	ContactOrganizationName	Contact Organization Name	NASDA,1401 OHASHI HATOYAMA-MACHI,HIKI-GUN,SAITAMA,350-0393,JAPAN,+81-492-98-1307,orderdesk@eoc.nasda.go.jp
18	StartOrbitNumber	Start Orbit Number	100
19	StopOrbitNumber	Stop Orbit Number	100
20	EquatorCrossingLongitude	Equator Crossing Longitude	-176.47
21	EquatorCrossingDate	Equator Crossing Date	2001-09-08
22	EquatorCrossingTime	Equator Crossing Time	23:01:21.87Z
23	OrbitDirection	Orbit Direction	DESCENDING or ACENDING
24	EphemerisGranulePointer	Ephemeris Granule Pointer	EL20010918
25	EphemerisType	Ephemeris Type	GPS
26	PlatformShortName	Platform Short Name	ADEOS-2 or EOS-PM1
27	SensorShortName	Sensor Short Name	AMSR or AMSR-E
28	NumberofScans	Number of Scan	834
29	NumberofMissingScans	Number of Missing Scan	0
30	ECSDataModel	ECS Data Model(name of metadata model)	B.0
31	DiscontinuityVirtualChannelCounter	Virtual channel Unit Counter Discontinuity	Discontinuation
32	QALocationPacket-Discontinuity	Packet Sequence Counter Discontinuity	Continuation
33	NumberofPackets	Number of Packets of L0 data	13344
34	NumberofInputFiles	Number of Input L0 Files	1
35	NumberMissingPackets	Number Missing Packets	0
36	NumberofGoodPackets	Number of Good Packets	13344
37	ReceivingCondition	Receiving Condition	Blank
38	EphemerisQA	Ephemeris limit check	OK
39	AutomaticQAFlag	Automatic QA Flag check	PASS
40	AutomaticQAFlagExplanation	Automatic QA Flag Explanation	1.MissingDataQA:Less than 1010 is available->OK, 2.AntennaRotationQA:Less than 20 is available->OK, 3.HotCalibrationSourceQA:Less than 20 is available->OK, 4.AttitudeDataQA:Less than 20 is available->OK,

Table 4.4.1-1 L1B Metadata

metadata index	metadata name	Description	metadata values (example)
			5.EphemerisDataQA:Less than 20 is available->OK, 6.QualityofGeometricInformationQA: Less than 0 is available->OK, 7.BrightnessTemperatureQA:Less than 20 is available->OK, All items are OK, 'PASS' is employed
41	ScienceQualityFlag	Science Data calculation Quality Flag	Blank
42	ScienceQualityFlagExplanation	Science Data calculation Quality Description	Blank
43	QAPercentMisssingData	QA Percent of Misssing Data	0
44	QAPercentOutofBoundsData	QA Percent Out of Bounds Data	0
45	QAPercentParityErrorData	Number of QA Percent Parity Error Data	0
46	ProcessingQADescription	Processing QA Description	PROC_COMP
47	ProcessingQAAttirbute	Processing QA Attribute Name	Automatic QA Flag etc
48	SatelliteOrbit	Satellite Orbit	Sun-synchronous_sub-recurrent
49	Altitude	Satellite Altitude	802.9km
50	OrbitSemiMajorAxis	Satellite Orbit Semi Major Axis	7181.317km
51	OrbitEccentricity	Satellite Orbit Eccentricity	0.00007
52	OrbitArgumentPerigee	Satellite Orbit Argument Perigee	244.018deg
53	OrbitInclination	Satellite Orbit Inclination	98.62deg
54	OrbitPeriod	Satellite Orbit Period	101minutes
55	RevisitTime	Revisit Time	4days
56	AMSR/AMSR-EChannel	AMSR/AMSR-E Channel	6.925GHz,10.65GHz,18.7GHz,23.8GHz,36.5GHz,50.3GHz,52.8GHz,89.0GHz-A,89.0GHz-B
57	AMSR/AMSR-EBandWidth	AMSR/AMSR-E Band Width	6G-350MHz,10G-100MHz,18G-200MHz,23G-400MHz,36G-1000MHz,50.3G-200MHz,52G-400MHz,89GA-3000MHz,89GB-3000MHz
58	AMSR/AMSR-EBeamWidth	AMSR/AMSR-E Beam Width	6G-1.8deg,10G-1.2deg,18G-0.64deg,23G-0.75deg,36G-0.35deg,50.3G-0.25deg,52G-0.25deg,89GA-0.15deg,89GB-0.15deg
59	OffNadir	Angle of offnadir	46.7deg : for 89GB 46.3deg
60	SpatialResolution(AzXEl)	Spatial Resolution	6G-39.8kmX69.5km,10G-26.6kmX46.3km,18G-14.4kmX25.1km,23G-16.6kmX28.9km,36G-7.7kmX13.5km,50.3G-5.5kmX9.6km,52G-5.5kmX9.6km,89GA-3.3kmX5.8km,89GB-3.3kmX5.7km
61	ScanningPeriod	Scanning Period	1.5sec
62	SwathWidth	Swath Width	1600km
63	DynamicRange	Dynamic Range	2.7K-340K
64	DataFormatType	Data Format Type	NCSA-HDF
65	HDFFormatVersion	HDF Format Version	Ver4.1r2
66	EllipsoidName	Ellipsoid Model Name	WGS84
67	SemiMajorAxisofEarth	Semi Major Axis of Earth	6378.1km
68	FlatteningRatioofEarth	Flattening Ratio of Earth	0.00335
69	SensorAlignment	Sensor Alignment	Rx=0.00000,Ry=0.00000,Rz=0.00000
70	ThermistorCountRangeWx	Thermistor Count Adaptable Range Wx	61,138,301,456,591,698,780,840,883,915,937,954,966,974,1024
71	ThermistorConversionTableWa	Thermistor Conversion Table Wa	0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
72	ThermistorConversionTableWb	Thermistor Conversion Table Wb	0.00000,0.06494,0.06135,0.06452,0.07407,0.09346,0.12195,0.16667,0.23256,0.31250,0.45455,0.58824,0.83333,1.25000,0.00000
73	ThermistorConversionTable	Thermistor Conversion Table Wc	-35.0000,-38.9610,-38.4663,-39.4194,-

Table 4.4.1-1 L1B Metadata

metadata index	metadata name	Description	metadata values (example)
	Wc		43.7778,-55.2336,-75.1220,-110.0000,-165.3488,-235.9375,-365.9091,-491.1765,-725.0000,-1127.5000,90.0000
74	ThermistorConversionTableWd	Thermistor Conversion Table Wd	0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
75	Platinum#1CountRangeWx	Platinum#1 Count Adaptable Range Wx	1168,1296,1536,1792,2032,2272,2512,2752,2992,3232,3472,3712,3952,4096
76	Platinum#1ConversionTableWa	Platinum#1 Conversion Table Wa	0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
77	Platinum#1ConversionTableWb	Platinum#1 Conversion Table Wb	0.00000,0.03906,0.04167,0.03906,0.04167,0.04167,0.04167,0.04167,0.04167,0.04167,0.04167,0.04167,0.04167,0.04167
78	Platinum#1ConversionTableWc	Platinum#1 Conversion Table Wc	-35.0000,-80.6250,-84.0000,-80.0000,-84.6667,-84.6667,-84.6667,-84.6667,-84.6667,-55.5000,-84.6667,-84.6667,-84.6667,-84.6667
79	Platinum#1ConversionTableWd	Platinum#1 Conversion Table Wd	0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
80	Platinum#2CountRangeWx	Platinum#2 Count Adaptable Range Wx	272,528,784,1040,1296,1536,1792,2032,2288,2528,2768,3008,3248,3472,3712,4096
81	Platinum#2ConversionTableWa	Platinum#2 Conversion Table Wa	0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
82	Platinum#2ConversionTableWb	Platinum#2 Conversion Table Wb	0.00000,0.07813,0.07813,0.07813,0.07813,0.08333,0.07813,0.08333,0.07813,0.08333,0.08333,0.08333,0.08333,0.08929,0.08333,0.00000
83	Platinum#2ConversionTableWc	Platinum#2 Conversion Table Wc	-140.0000,-161.2500,-161.2500,-161.2500,-161.2500,-168.0000,-160.0000,-169.3333,-158.7500,-170.6667,-170.6667,-170.6667,-170.6667,-190.0000,-169.3333,140.0000
84	Platinum#2ConversionTableWd	Platinum#2 Conversion Table Wd	0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
85	Platinum#3CountRangeWx	Platinum#3 Count Adaptable Range Wx	368,704,1040,1360,1696,2032,2352,2688,3008,3344,3664,4000,4096
86	Platinum#3ConversionTableWa	Platinum#3 Conversion Table Wa	0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
87	Platinum#3ConversionTableWb	Platinum#3 Conversion Table Wb	0.00000,0.00893,0.00893,0.00938,0.00893,0.00893,0.00938,0.00893,0.00938,0.00893,0.00938,0.00893,0.00000
88	Platinum#3ConversionTableWc	Platinum#3 Conversion Table Wc	12.0000,8.7143,8.7143,8.2500,8.8571,8.8571,7.9500,9.0000,7.8000,9.1429,7.6500,9.2857,45.0000
89	Platinum#3ConversionTableWd	Platinum#3 Conversion Table Wd	0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0 fixed value
90	CoefficientAvv	Brightness Temperature Coefficient Avv	6G-1.000,10G-1.000,18G-1.000,23G-1.000,36G-1.000,50G-1.000,89GA-1.000,89GB-1.000
91	CoefiicientAhv	Brightness Temperature Coefficient Ahv	6G-0.000,10G-0.000,18G-0.000,23G-0.000,36G-0.000,50G-0.000,89GA-0.000,89GB-0.000
92	CoefficientAov	Brightness Temperature Coefficient Aov	6G-0.000,10G-0.000,18G-0.000,23G-0.000,36G-0.000,50G-0.000,89GA-0.000,89GB-0.000

Table 4.4.1-1 L1B Metadata

metadata index	metadata name	Description	metadata values (example)
93	CoefficientAhh	Brightness Temperature Coefficient Ahh	6G-1.000,10G-1.000,18G-1.000,23G-1.000,36G-1.000,52G-1.000,89GA-1.000,89GB-1.000
94	CoefficientAvh	Brightness Temperature Coefficient Avh	6G-0.000,10G-0.000,18G-0.000,23G-0.000,36G-0.000,52G-0.000,89GA-0.000,89GB-0.000
95	CoefficientAoh	Brightness Temperature Coefficient Aoh	6G-0.000,10G-0.000,18G-0.000,23G-0.000,36G-0.000,52G-0.000,89GA-0.000,89GB-0.000
96	CSM Temperature	Temperature of Cosmic Microwave Background(CSB) on cold sky mirror	6GV-3.390,6GH-3.390, 10GV-3.040,10GH-3.040, 18GV-3.040,18GH-3.040, 23GV-3.040,23GH-3.040, 36GV-3.040,36GH-3.040, 50GV-3.040,52GV-3.040, 89GAV-3.040,89GAH-3.040, 89GBV-3.040,89GBH-3.040
97	CoRegistrationParameterA1	CoRegistrationParameterA1	6G-0.57534,10G-0.87671,18G-0.72603, 23G-0.46575,36G-0.47945,50G-0.00000
98	CoRegistrationParameterA2	CoRegistrationParameterA2	6G--0.23288,10G--0.17323,18G-0.06849,23G--0.19178,36G-0.00000,50G-0.00000
99	CalibrationMethod	The method name of HTS correction	RxTemperatureReferenced,SpillOver,MoonLightEffect
100	HTSCorrectionParameterVersion	The version number of the parameter file used for HTS correction	ver0001
101	SpillOverParameterVersion	The version number of the parameter file used for the spill over correction	ver0001
102	MoonLightEffectParameterVersion	The version number of the parameter file used for eliminating moon light effect on 10～89GHz CSM data	ver0001

4.4.2 L2 Metadata

Table 4.4.2-1 L2 Metadata

metadata index	metadata name	Description	metadata values (example)
0	ShortName	Product name	AMSR-L2
1	GeophysicalName	Geophysical quantity name	Water Vapor
2	VersionID	ID of product version	0-255
3	SizeMBECSDataGranule	Product size (Mbyte)	30(actual)
4	Local Granule ID	Number for production management	A2AMS020101001A_P2WV0Tak111
5	ProcessingLevelID	ID of processing level	L2
6	ProductionDateTime	Time of production (UT)	2002-1-3-T00:00:00.00Z
7	RangeBeginningTime	Time to start observing (UT)	00:00:00.00Z
8	RangeBeginningDate	Date to start observing (UT)	2002-1-3
9	RangeEndingTime	Time to end observing (UT)	01:00:00.00Z
10	RangeEndingDate	Date to end observing (UT)	2002-1-3
11	GringPointLatitude	Area of interest for latitude	90
12	GringPointLongitude	Area of interest for longitude	-180
13	PGEName	Name of software	(max 20 character)
14	PGEVersion	Version of software	(max 18 character)
15	PGEAlgorismDeveloper	Name of algorism developer	(max 20 character)
16	InputPointer	Input file name	A2AMS02010101MD_P01B0000000000.00
17	ProcessingCenter	Name of data processing center	HATOYAMA
18	ContactOrganizationName	Organization name to contact about this product	NASDA Address:

Table 4.4.2-1 L2 Metadata

metadata index	metadata name	Description	metadata values (example)
			OOAZA-OHASHI-AZA-NUMANOUE HIKI-GUN SAITAMA,JAPAN Postal code : 350-0393 Telephone Number : 0492-98-1200 E-mail Address : abc@rd.tksc.nasda.go.jp Instructions : 9:20(JST) - 17 (JST) is the working time
19	StartOrbitNumber	Start orbit number	100
20	StopOrbitNumber	Stop orbit number	100
21	EquatorCrossingLongitude	Equator crossing latitude	89
22	EquatorCrossingDate	Equator crossing date	1998.2.4
23	EquatorCrossingTime	Equator crossing time	00:30:00Z
24	OrbitDirection	Orbit direction	DESCENDING
25	EphemerisGranulePointer	File name for using orbit	EPHEMERIS-1
26	EphemerisType	Type of using orbit	GPS
27	PlatformShortName	Abbreviated name of platform	ADEOS-II
28	SensorShortName	Abbreviated name of observing sensor	AMSR
29	NumberofScan	Number of scan	2020
30	ECSDataModel	Name of meta data model	B.0
31	DiscontinuityVirtualChannelCounter	Discontinuity flag of virtual channel unit counter	Continuation/Discontinuation
32	QALocationPacketDiscontinuity	Discontinuity flag of packet sequence counter	Continuation/Discontinuation
33	NumberofPackets	Number of L0 packet	32320
34	NumberofInputFiles	Number of L0 file	1
35	NumberofMissingPackets	Number of missing packet	nnnn
36	NumberofGoodPackets	Number of good packet	nnnn
37	ReceivingCondition	Condition for record or receive	GOOD
38	EphemerisQA	Result of limit check for ephemeris	OK
39	AutomaticQAFlag	Result by program check	PASS
40	AutomaticQAFlagExplanation	Explanation of program check	
41	ScienceQualityFlag	Flag when it calculates geophysical quantity	Blank for L1A,L1B,L1BMap
42	ScienceQualityFlagExplanation	Explanation when it calculate geophysical quantity	Blank for L1A,L1B,L1BMap
43	QAPercentMissingData	Number of missing data	nnn
44	QAPercentOutofBoundsData	Ratio of data out of bound	nnn

4.4.3 L3 Metadata

Table 4.4.3-1 L3 Metadata

metadata index	metadata name	Description	metadata values (example)
0	Short Name	Product name	AMSR-L3
1	GeophysicalName	Geophysical quantity name	Water Vapor,
2	VersionID	ID of product version	0-255
3	SizeMBECSDataGranule	Product size (Mbyte)	30(actual)
4	Local Granule ID	Number for production management	A2AMS010101A_P3WV0Tak111E0
5	ProcessingLevelID	ID of processing level	L3
6	ProductionDateTime	Time of production (UT)	2002-1-3-T00:00:00.00Z
7	RangeBeginningTime	Time to start observing (UT)	00:00:00.00Z
8	RangeBeginningDate	Date to start observing (UT)	2002-1-3
9	RangeEndingTime	Time to end observing (UT)	01:00:00.00Z
10	RangeEndingDate	Date to end observing (UT)	2002-1-3
11	InputPointer	Name of algorism developer	A2AMS02010101MD_P01B0000000000.00

Table 4.4.3-1 L3 Metadata

metadata index	metadata name	Description	metadata values (example)
12	StartOrbitNumber	Start orbit number	100
13	StopOrbitNumber	Stop orbit number	100
14	OrbitDirection	Orbit direction	DESCENDING
15	PlatformShortName	Abbreviated name of platform	ADEOS-II
16	SensorShortName	Abbreviated name of observing sensor	AMSR
17	ECSDataModel	Name of meta data model	B.0
18	PGEName	Name of software	(max 20 character)
19	PGEVersion	Version of software	(max 18 character)
20	ProcessingCenter	Name of data processing center	HATOYAMA
21	ContactOrganizationName	Organization name to contact about this product	

5 Sample Program List

This chapter provides sample programs for outputting the value of data stored in L1B, L2, and L3 products. Sample programs and data are stored in "*sample*" directory, which is in the *ADIT installation directory*. Its structure is shown in Fig. 5-1. The way of program compilation is shown in the header part of each source code (refer to Fig. 5-2). More detail is described in Section 3.3.2 and Section 3.3.3.

An executable object file will be produced after the compilation according to the above. Its name is extracted from the file name except its extension. It is different to invoke the process for C and Fortran code.

【In the case of C program】

% Executable file name△Input data name

△ denotes blank.

e.g.)The following shows how to invoke the executable object file which is generated from L1_swath1b.c. Suppose *L1_swath1b* is an executable file name and *P1AME030609207MA_P01B0000000.00.sample* is an input data name, it is shown as below.

% L1_swath1b P1AME030609207MA_P01B0000000.00.sample

【In the case of Fortran program】

% Executable file name

＊ As an input data file name cannot be specified as a parameter for the execution of Fortran program, it shall be specified as a statement in the codes.

e.g.)The following shows how to invoke the executable object file which is generated from L1_swath1b.f.

% L1_swath1b

```
…【sample】┬ 「readme.txt」      (Explanation about sample program)
          ├【C】                (C sample program storing directory)
          ├【Fortran】          (Fortran sample program storing directory)
          ├【Fortran_for_Linux】
          │   (Fortran sample program for linux storing directory)
          └【data】             (Sample data storing directory)
```

Sample data is stored only when ADIT with sample data is downloaded.

Fig. 5-1 Directory Structure

```
/************************************************************************
  This is a sample program to read AMSR/L1B data, and
  following are instructions for compiling a sample program with ADIT.

      For SGI
            cc  -DSGI -xansi -O -s -o L1_cal L1_cal.c ¥
               -I$HDFINC -I../../include ¥
            -L$HDFLIB -L../../lib ¥
            -lADIT -lmfhdf -ldf [-ljpeg] -lz -lm

      For SunOS
            cc -DSUN -Xc -xO2 -lnsl -o L1_cal L1_cal.c ¥
                  -I$HDFINC -I../../include ¥
                  -L$HDFLIB -L../../lib ¥
            -lADIT -lmfhdf -ldf [-ljpeg] -lz -lm

      For HP-UX
            cc -DHP9000 -Ae -s -o L1_cal L1_cal.c ¥
                  -I$HDFINC -I../../include ¥
                  -L$HDFLIB -L../../lib ¥
            -lADIT -lmfhdf -l   df [-ljpeg] -lz -lm

      FOR DEC ALPHA
            cc -DDEC_ALPHA -Olimit 2048 -std1 -o L1_cal L1_cal.c ¥
                  -I$HDFINC -I../../include ¥
                  -L$HDFLIB -L../../lib ¥
            -lADIT -lmfhdf -ldf [-ljpeg] -lz -lm

      FOR LINUX
            gcc -DLINUX -ansi -o L1_cal L1_cal.c ¥
                  -I$HDFINC -I../../include ¥
                  -L$HDFLIB -L../../lib ¥
            -lADIT -lmfhdf -ldf [-ljpeg] -lz -lm

Note:
      $HDFINC indicates the directory of included files of HDF liblary.
      $HDFLIB indicates the directory of library files of HDF liblary.

************************************************************************/
```

The way of compilation for some typical computers is described at the header part of each sample program.

Fig. 5-2 Description for compile

```
program main

include 'AMSR_f.h'

character*46 fname
data fname/' ../data/P1AME030609207MA_P01B0000000.00.sample     '/
integer status
integer i
integer sd_id

record /CAL/          cal
          .
          .
          .
          .
```

Specifying the input data

Fig. 5-3 Specifying the input data in Fortran program

Table 5-1 Sample program list

No.	Program file name	Explanation
1.	(read a scan) L1_swath1b.c (C) L1_swath1b.f (Fortran) (read a number of scans) L1_swath1b_line.c (C) L1_swath1b_line.f (Fortran)	This code describes how to display the following data on a screen. This is applicable for a L1B product. ・ Scan_Time ・ Brightness_Temperature(6GHz-89GHz) ・ Lat_of_Observation_Point_Except_89B ・ Long_of_Observation_Point_Except_89B ・ Lat_of_Observation_Point_for_89B ・ Long_of_Observation_Point_for_89B
2.	(read a scan) L1_sunearth.c (C) L1_sunearth.f (Fortran) (read a number of scans) L1_sunearth_line.c (C) L1_sunearth_line.f (Fortran)	This code describes how to display the following data on a screen. This is applicable for a L1B product. ・ Sun_Azimuth ・ Sun_Elevation ・ Earth_Incidence ・ Earth_Azimuth ・ Land/Ocean_Flag_for_6_10_18_23_36_50_89A
3.	(read a scan) L1_status1b.c (C) L1_status1b.f (Fortran) (read a number of scans) L1_status1b_line.c (C) L1_status1b_line.f (Fortran)	This code describes how to display the following data on a screen. This is applicable for a L1B product. ・ Position_in_Orbit ・ Data_Quality
4.	(read a scan) L1_cal.c (C) L1_cal.f (Fortran)	This code describes how to display the following data on a screen. This is applicable for a L1B product. ・ Hot_Load_Count_6_to_52

Table 5-1 Sample program list

No.	Program file name	Explanation
	(read a number of scans) L1_cal_line.c (C) L1_cal_line.f (Fortran)	・ Hot_Load_Count_89 ・ Cold_Sky_Mirror_Count_6_to_52 ・ Cold_Sky_Mirror_Count_89 ・ Antenna_Temp_Coef(Of+Sl) ・ Rx_Offset/Gain_Count ・ SPC_Temperature_Count ・ SPS_Temperature_Count
5.	(read a scan) L1_navi.c (C) L1_navi.f (Fortran) (read a number of scans) L1_navi_line.c (C) L1_navi_line.f (Fortran)	This code describes how to display the following data on a screen. This is applicable for a L1B product. ・ Navigation_Data ・ Attitude_Data
6.	(read a scan) L1_scantime.c (C) L1_scantime.f (Fortan) (read a number of scans) L1_scantime_line.c (C) L1_scantime_line.f (Fortan)	This code describes how to display the following data on a screen. This is applicable for a L1B product. ・ Scan_Time
7.	L1_89GHz_low.c(C) L1_89GHz_low.f(Fortran)	This code describes how to display the following data on a screen. This is applicable for a L1B product. ・89GHz lof frequency data
8.	(read a scan) L2_swath2.c (C) L2_swath2.f (Fortarn) (read a number of scans) L2_swath2_line.c (C) L2_swath2_line.f (Fortarn)	This code describes how to display the following data on a screen. This is applicable for a L2 product. ・ Scan Time Table ・ Geophysical Quantity Data ・ Lat. of observation point except 89B ・ Long. of observation point except 89B
9.	(read a scan) L2_status2.c (C) L2_status2.f (Fortran) (read a number of scans) L2_status2_line.c (C) L2_status2_line.f (Fortran)	This code describes how to display the following data on a screen. This is applicable for a L2 product. ・ Position_in_Orbit ・ Data Quality
10.	(read a scan) L2_scantime.c (C) L2_scantime.f (Fortarn) (read a number of scans) L2_scantime_line.c (C) L2_scantime_line.f (Fortarn)	This code describes how to display the following data on a screen. This is applicable for a L2 product. ・ Scan Time Table
11.	L3.c (C) L3.f (Fortran)	This code describes how to display the following data on a screen. This is applicable for a L3 product.

Table 5-1 Sample program list

No.	Program file name	Explanation
		・ Mean for Brightness Temperature(6GHz-89GHz) ・ Mean for Geophysical Data
12.	sample1.c (C) sample1.f (Fortran)	This code describes how to set meta data and each data set in structures. This is applicable for a L1B product. More detail is in Section 3.2.3 and 3.3.3.
13.	sample2.c (C) sample2.f (Fortran)	This code describes how to set meta data and each data set in structures This is applicable for a L2 product.
14.	sample3.c (C) sample3.f (Fortran)	This code describes how to set meta data and each data set in structures. This is applicable for a L3 product.

Table 5-2 Sample data list

No.	Data file name	Explanation
1.	P1AME030201006MA_P01B0000000.00.sample	AMSR-E L1B product
2.	P1AME030609207A_P2WV0Tak071.00.sample	AMSR-E L2 product
3.	P1AME030609A_P3WV0Tak071E0.00.sample	AMSR-E L3 product