GCOM-W1
AMSR2 Product I/O Tool Kit (AMTK)
Operation manual

Change record

Issue	Date	Sheet	Description of change
NC	2010/3	—	—
1.11	2013/01/24	—	First Release
1.12	2013/05/13	—	Fixed typo
1.13	2013/12/03	—	To change the scale factor of Precipitation. 0.1→0.01
1.14	2015/03/26	—	Revise the definition of the missing value. Revise the definition of the AM2_DEF_RMISS value. -9999.00->-9999.99

Contents

1 HDF library and AMSR2 I/O Tool kit

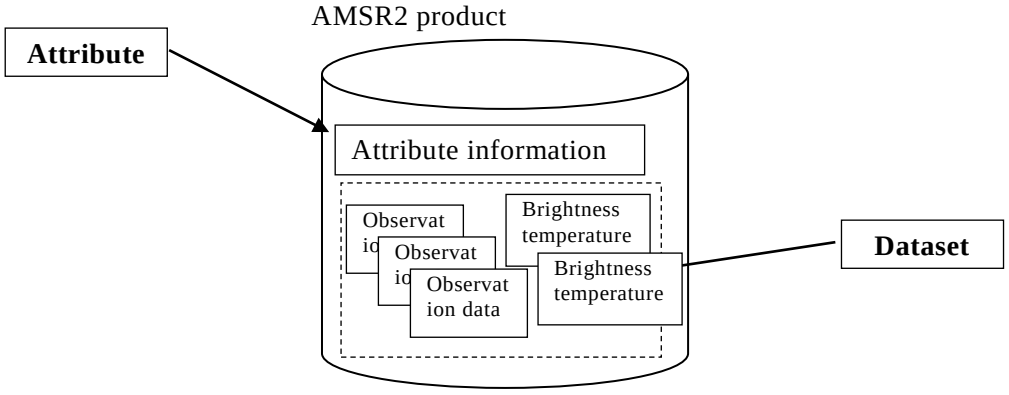
AMSR2 I/O Tool kit(AMTK)is offered to use AMSR2 product data, which is the data of Global Change Observation Mission 1st –Water, with a software program developed by C-language or FORTRAN. AMSER2 product is classified into Level 1 to 3 and the data stored in the data format HDF (Hierarchical Data Format). AMTK can be easily accessed to the AMSR2 product data of HDF.

1.1 About HDF

All of the AMSR2 product data are stored in the data format of HDF.

HDF is the data format which doesn't depend on the user's computer compositio n developed by NCSA (The National Center for Supercomputing Applications: Illin ois University) to use the information. The HDF file has the HDF4 form and the HDF5 form, and HDF5 completely reviews the problem of HDF4 (data type of t wo or more kinds of that there is a limitation in the data size etc.) and is being offered. The file format of HDF5 is applied in the AMSR2 product. AMTK access es the file of HDF5, and acquires (input function) or sets (output function) inform ation specified by the user program.

The HDF file is divided into Attribute part containing attribute information and Dataset part containing product information. The Attribute part of AMSR2 produ ct contains the metta data and the Dataset part contains the product data such a s observation data and latitude longitude data.



HDF5

図 1.1-1 HDF File composition

1.2 About AMTK

AMTK is a toolkit developed to access the AMSR2 product data, the DEM file, and the weather file easily on C language and the Fortran language program. AMTK is composed of the I/O function group of AMSR2 product information.

The way to access to HDF is to open the HDF file, specify identifier (hid_t) returned by this opening, and input and output information on Attribute and each Dataset. Dataset is specified by using the HDF identification number (Please refer to Chapter 6.1.1 for details) to identify two or more Dataset in AMTK.

(1) Requirements

AMTK runs under the environment shown in table 1.2-1.

Table 1.2-1 Requirements

Item	DELL4		Sun-Fire-V440	SGI Origin2000
OS	Red Hat Enterprise Linux Client release 5 (Tikanga)		SunOS 5.9	IRIX64 6.5
compiler	C Language	Fortran	Sun Studio 11	MIPSpro Compilers: Version 7.30
	gcc, g++ Intel C++ Compiler	g77, g95 (*1) f77, f95 Intel Fortran Compiler PGI Fortran Compiler		
RAM	12GB		8GB	1GB
HDF Library	HDF5-1.8.4-patch1 (*2)			

(*1)AMTK using g95 compiler is shown below website.

Linux x86_64/EMT64 (32 bit D.I.) Default integer of 32 bits, compatible with older programs

(*2)Sun-Fire-V440 is compiled from source code.

You need to make changes to some parts of source code of HDF library before compile SGI Origin2000. Refer to the 2.1 Installing HDF library section.

(2) Platform

AMTK supports both 32 / 64 bits machines.

(3) Language and Compiler

AMTK is used with language and compiler to show in Table 1.2-2

Table 1.2-2 Language and Compiler

OS	C-Language	FORTRAN	
		Fortran77	Fortran95
RedHat Enterprise Linux	gcc, g++	g77, f77	g95, f95
	Intel C++ Compiler	Intel Fortran Compiler	
		PGI FORTRAN77	PGI FORTRAN90/95
IRIX64	IRIX CC	IRX FORTRAN77	MIPSpro Compilers:Version 7.30

Solaris Sun OS	Sun Studio 11 C	Sun Studio 11	Sun Studio 11 Fortran

2 Installing HDF library and AMTK

The AMSR2 product data is made by using Release 1.8.4 Patch1 in the HDF5 l ibrary. Here, it explains the installation of Release 1.8.4 Patch1 in the HDF5 libr ary.

2.1 Installing HDF library

There are two kinds of methods to install the HDF library. One is to install the compiled binary data of Release 1.8.4 Patch1 and the other is to install the source code of the HDF library.

2.1.1 Getting HDF5 library

The HDF5 library is obtained from the homepage of the HDF group. The homepage of the latest HDF5 library version is shown below.

http://www.hdfgroup.org/HDF5/release/obtain5.html#obtain

The table shows the object of the download in March, 2010. Please obtain the HDF5 library corresponding to computer and OS used.

Table 2.1-1 HDF5 library

Platform	Download File name	Remark
All Platform	src/hdf5-1.8.4-patch1.tar.gz	source code
Linux 2.6 i686	hdf5-1.8.4-patch1-linux-shared.tar.gz	Binary(common)
Linux 2.6 x86_64	hdf5-1.8.4-patch1-linux-x86_64-shared.tar.gz	Binary(common)
Solaris 2.10 (32-bit)	hdf5-1.8.4-patch1-solaris-shared.tar.gz	binary(common)

Because the sz library is used in HDF5, it is necessary to obtain it for szip-2.1.tar.gz separately.

Talbe 2.1-2 sz library

At the obtaining destination	Remark
ftp://ftp.hdfgroup.org/lib-external/szip/2.1/src/	

2.1.2 Installing HDF5 (sz library)

Here, it explains for the platform of Linux2.6.

(1) Getting library

Get the szip-2.1.tar.gz file and Linux 2.6 i686(hdf5-1.8.4-patch1-linux-shared.tar.gz).

(2) Installing SZ library

Uncompress the szip-2.1.tar.gz file and install. The example of installing the sz library in/usr/local/lib is shown as follows.

```
$ tar xzf szip-2.1.tar.gz
$ cd szip-2.1/
$ ./configure –prefix=/usr/local
$ make
```

Need to be super user!

```
# make install
```

2.1.3 Installing HDF library(binary)

Uncompress hdf5-1.8.4-patch1-linux-shared.tar.gz

```
$ tar xzf hdf5-1.8.4-patch1-linux-shared.tar.gz
```

When the above command is executed, the directory named hdf5-1.8.4-patch1-linux-shared is made. The file and the directory shown in Table 2.2-1 are deployed in this directory.

Table 2.1-1 HDF library

Name	file or directory	Contents	Remark
COPYING	File	It is Copyright Notice	
README	File	Easy use and sz library is explained.	
RELEASE.txt	File	It is release memo.	
bin	Directory	The directory deployed HDF tool	
include	Directory	The directory deployed include files	
lib	Directory	The directory deployed library	

The file related to HDF is not cared about in any directory because the include file of HDF and the directory of the library are specified with the installer of AMTK. To put it together on the Description of the installation of AMTK, it will put it on the directory of/HDF5/shared here.

```
# mkdir /HDF5
# mkdir shared
```
This directory must be made by super user.

```
$ cd hdf5-1.8.4-patch1-linux-shared
$ cp –r include/ /HDF5/shared
$ cp –r lib/ /HDF5/shared
```
Copy HDF related files

2.1.4 Installing HDF library(source code base)

Uncompress hdf5-1.8.4-patch1.tar.gz.

```
$ tar xzf hdf5-1.8.4-patch1.tar.gz
```

For example, install HDF5 library to "/HDF5/shared" directory.

```
# cd hdf5-1.8.4-patch1
# ./configure --prefix=/HDF5/shared
# make
# make install
```
Set install directory. And compile HDF5 library.

○ Installation of SGI environment

When you install HDF5 library in SGI(IRIX64 / MIPSpro Compilers) environm ent, you may fault installtion as shown below.

```
92 errors detected in the compilation of "h5tools.c".
*** Error code 1 (bu21)
*** Error code 1 (bu21)
*** Error code 1 (bu21)
```
These messages mean compiling error.

In this case, this probrem may clear with a modification.

1) Edit the hdf5-1.8.4-patch1/tools/lib/h5tools_error.h file with any editer applic ation.
2) Modify the sentence as below in 35 lines in the file.

・**Before**: #error "We need __func__ or __FUNCTION__ to test function names! "

・**After** : #define "We need __func__ or __FUNCTION__ to test function names! "

3) Compile as below command.

```
# ./configure --prefix=/HDF5/shared CFLAGS=-64
# make
# make install
```

Set install directory and add “-64” option in CFLAGS.

2.2 Installing AMTK

2.2.1 Installing AMTK on Linux

(1) Development of file

Uncompress AMTK_AMSR2_Ver1.13.tar.gz file and deploy the files

```
$ tar xzf AMTK_AMSR2_Ver1.13.tar.gz
```

When the above command is executed, the directory named AMTK_AMSR2 is m ade. In this directory, it is Table 2.2 The file and the directory of three are prog ressed.

Table 2.2-1 The comtents in AMTK

Name	file or directory	Contents	Remark
Makefile.in autom4te.cash config.gess config.sub configure configure.in install-sh	File	It is installer for AMTK	
include/	Directory	It is a directory deployed the in clude file of AMTK.	
lib/	Directory	It is a directory deployed the in clude file of AMTK library.	
src/	Directory	It is a directory deployed the in clude file of AMTK source cod e.	
sample/	Directory	It is a directory deployed the in clude file of sample program.	The leap second file is stored. sample/data/leapsec.dat
share/	Directory	Common configure files are stor ed.	Refer to the 2.3 section.

(2) Installing library

After it moves to the directory of AMTK, the installer of AMTK (configure com mand) is executed. In the installer, Makefile that suits the machine environment is generated.

```
$ ./configure
```

If it is an installation of usual HDF, it is not necessary to specify it because t he file

related to following HDF is retrieved with the installer.

○HDF include file.

Hdf5.h under /*/*/include or /*/*/*/include

If you installed it to arbitrary directory, set below commands.

○HDF library file

```
--with-hdf-lib=/HDF5/shread/lib
```

○HDF include file

```
--with-hdf-include=/HDF5/shread/include
```

Afterwards, the make command is executed by using generated Makefile.

```
$ make
```

The example of the continuation of the configure command and the make comm and execution is shown.

```
$ ./configure
checking build system type... i686-redhat-linux-gnu
checking host system type... i686-redhat-linux-gnu
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ANSI C... none needed
/HDF5/shared/lib
/HDF5/shared/include
configure: creating ./config.status
config.status: creating Makefile
```

(run compile)

```
$ make
gcc -c src/amtk_get.c -g -DDBG -DLINUX -ansi -g -DDBG -o src/amtk_get.o    -I./include
-I/HDF5/shared/include -I./src
gcc -c src/amtk_set.c -g -DDBG -DLINUX -ansi -g -DDBG -o src/amtk_set.o    -I./include
-I/HDF5/shared/include -I./src
gcc -c src/amtk_hdf.c -g -DDBG -DLINUX -ansi -g -DDBG -o src/amtk_hdf.o    -I./include
-I/HDF5/shared/include -I./src
gcc -c src/amtk_latlon.c -g -DDBG -DLINUX -ansi -g -DDBG -o src/amtk_latlon.o    -I./include
-I/HDF5/shared/include -I./src
gcc -c src/amtk_scantime.c -g -DDBG -DLINUX -ansi -g -DDBG -o src/amtk_scantime.o    -I./include
-I/HDF5/shared/include -I./src
gcc -c src/IOTK_common.c -g -DDBG -DLINUX -ansi -g -DDBG -o src/IOTK_common.o    -I./include
-I/HDF5/shared/include -I./src
ar
cr ./lib/libAMSR2.a ./src/amtk_get.o ./src/amtk_set.o ./src/amtk_hdf.o ./src/amtk_latlon.o ./src/amtk_scantime.o
  ./src/IOTK_common.o
rm ./src/*.o
```

The library making is confirmed. Making the AMTK library ends if lib directory under libAMSR2.a is made.

```
$ ls –l /home/amtk/AMTK /lib
Total 208
-rw-r--r-- 1 amtk amtk 204672    25 July 19:01 libAMSR2.a
```

2.2.2 Installing AMTK on UNIX

(1) Development of file

Uncompress AMTK_AMSR2_Ver1.12.tar.gz file and deploy the files

```
$ tar xzf AMTK_AMSR2_Ver1.12.tar.gz
```

When the above command is executed, the directory named AMTK_AMSR2 is m ade. In this directory, it is Table 2.3 the file and the directory of three are progr essed.

Table 2.2-3 The comtents in AMTK

Name	file or directory	Contents	Remark
Makefile.in autom4te.cash config.gess config.sub configure configure.in install-sh	File	It is installer for AMTK	
include/	Directory	It is a directory deployed the include file of AMTK.	
lib/	Directory	It is a directory deployed the include file of AMTK library.	
src/	Directory	It is a directory deployed the include file of AMTK source code.	
sample/	Directory	It is a directory deployed the include file of sample program.	The leap second file is stored. sample/data/leapsec.dat
share/	Directory	Common configure files are stored.	Refer to the 2.3 section.

(2) Installing library

After it moves to the directory of AMTK, Makefile that suits the machine environment is made. Makefile.in is copied, and the model of Makefile is made.

```
# cp Makefile.in Makefile
```

Please change the following items of Makefile in the editor.

- CC

```
CC= cc
```

- CFLAG

```
CFLAGS=    -DSGI –xansi –O –s -64                    in the case of SGI
```

```
CFLAGS=    - DSunOS –Xc –xO2 –lnsl                   in the case of Sun
```

- HDF Directory

```
Directory where HDFINC=HDF include file exists
```

The make command is executed. The execution example is shown in the following.

(run compile)

```
$ make
cc -c src/amtk_get.c -DSunOS -xO2 -lnsl -o src/amtk_get.o -I./include -I/HDF5/shared/include -I./src
cc -c src/amtk_set.c -DSunOS -xO2 -lnsl -o src/amtk_set.o -I./include -I/HDF5/shared/include -I./src
cc -c src/amtk_hdf.c -DSunOS -xO2 -lnsl -o src/amtk_hdf.o -I./include -I/HDF5/shared/include -I./src
cc -c src/amtk_latlon.c -DSunOS -xO2 -lnsl -o src/amtk_latlon.o -I./include -I/HDF5/shared/include -I./src
cc -c src/amtk_scantime.c -DSunOS -xO2 -lnsl -o src/amtk_scantime.o -I./include -I/HDF5/shared/include
-I./src
cc -c src/IOTK_common.c -DSunOS -xO2 -lnsl -o src/IOTK_common.o -I./include -I/HDF5/shared/include
-I./src
ar
cr ./lib/libAMSR2.a ./src/amtk_get.o ./src/amtk_set.o ./src/amtk_hdf.o ./src/amtk_latlon.o ./src/amtk_scantime.o
  ./src/IOTK_common.o
rm ./src/*.o
```

The library making is confirmed. Making the AMTK library ends if lib directory under libAMSR2.a is made.

```
$ ls –l /home/amtk/AMTK /lib
Total 208
-rw-r--r-- 1 amtk amtk 204672    25 July    19:01 libAMSR2.a
```

2.3 Execution environment setting of AMTK

The setting of the environment variable is needed for the execution of the application using the AMTK library. Table 2.3-1 shows the list of the environment parameter.

Table 2.3-1 AMTK environment parameter

Name	Description	Remarks
LD_LIBRARY_PATH	Passing a shared library of HDF is specified.	
L3LATLONFILEDIR	The directory where the latitude longitude file of L3 exists is specified.	
AMSR2_LEAP_DATA	The directory where the leap second file exists is specified. The directory where the leap second file exists is specified.	sample/data/leapsec.dat
GEOPHYSICALFILE	The directory where the geophysical quantity definition is specified.	share/data/geophysiocal_file

As for the environment parameter, specification is different according to the shell used. Every time, it will not be necessary to set it if it sets it to the log in sh

ell though the environment parameter can be individually set by the command w hen executing it. Here, the method of specification for the log in shell is shown.

(1) For csh

Please add the following to the ".cshrc" file that exists in the home directory.

```
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:/HDF5/shared/lib:/usr/local/lib
setenv L3LATLONFILEDIR (Directory where latitude longitude file of L3 exists)
setenv AMSR2_LEAP_DATA (Directory where leap second file exists)
setenv GEOPHYSICALFILE (Geophysical quantity definition sile exists)
```

(2) For bash

Please add the following to the ".bashrc" file that exists in the home directory.

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/HDF5/shared/lib:/usr/local/lib
export L3LATLONFILEDIR= (Directory where latitude longitude file of L3 exists)
export AMSR2_LEAP_DATA= (Directory where leap second file exists)
export GEOPHYSICALFILE (Geophysical quantity definition sile exists)
```

2.4 About leap second file

Leap second file (sample/data/leapsec.dat) in the AMTK library is a setting as o f August, 2010. Thereafter, when the leap second is updated, it is necessary to re place the leap second file.

2.5 Geophysical quantity definition file

It defines the property of the geophysical quantity of L2/L3.

Modification of this file enables to add or modify the definition of the geophysical quantity without modificationof AMTK own

The Geophysical quantity definition file is written in the format shown below.

Table 2.5-1 Geophysical quantity definition file format

Format by line		Geophysical Name,Scale Factor,Unit,EQR,PS-N,PS-S
1	Geophysical Name	Name of geophysical quantity. It is the name corresponding Metadata ”Geo physicalName”. (within 36 ASCII characters)
2	Scale Factor	Scale factor affects to the data set “GeophysicalData. (within 6 numberic characters and decimal point)
3	Unit	Unit affects to the data set “GeophysicalData” (within 10 ASCII characters)
4	EQR	Identification mark of Lat/Long file in case of Projection is “EQR”. This is used to identify L3 Lat/Long file depending Geophysical quantity. ("EQR" / "-")
5	PS-N	Identification mark of Lat/Long file in case of Projection is “PS-N”.

		This is used to identify L3 Lat/Long file depending Geophysical quantity. ("SIC" / "SND" / "-")
6	PS-S	Identification mark of Lat/Long file in case of Projection is “PS-S”. This is used to identify L3 Lat/Long file depending Geophysical quantity. ("SIC" / "SND" / "-")

- One geophysical quantity is defined by one line.
- Each data are divided with a comma-delimited (‘,‘).
- It set a hyphen (‘-‘) in the data item that does not need the setting.
- The hash mark (‘#’) at the beginning of line is used as simble of comment l ine. It is excluded by a reading object.
- The Brightness Temperature is defined in this file. In case of the brightness temperature is defined, it is not necessary to set the ScaleFactor and Unit.

【the example of Geophysical quantitiy file】

```
#Geophysical Name,Scale Factor,Unit,EQR,PS-N,PS-S      (comment line)
Total Precipitable Water,0.01,Kg/m2,EQR,-,-
Sea Ice Concentration,0.1,%,-,SIC,SIC
Snow Depth,0.1,cm,EQR,SND,-
Brightness Temperature (6GHz),-,-,EQR,SIC,SIC
```

In the default Geophysical quantity file (share/data/geophysiocal_file) which AMTK offers, the value should be set in Metadata “GeophysicalName”which is necessa ry for HDF access is described.

The contents of Geophysical quantity file corresponding to the value of Metadat a “GeophysicalName” are shown on the following.

Table 2.5-2 Geophysical quantity file

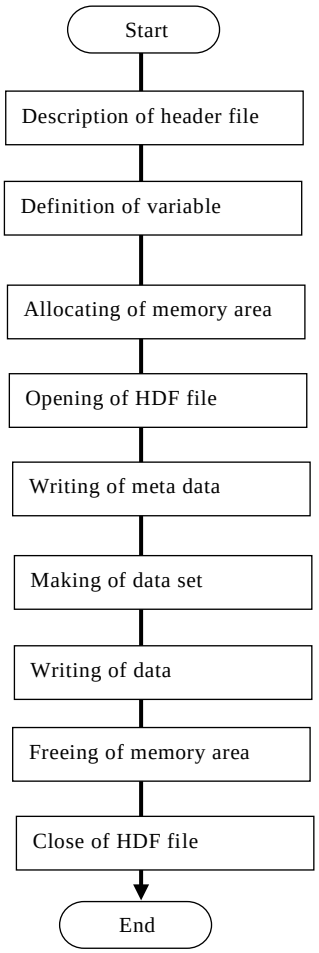
GeophysicalName	Unit	Scale Factor	EQR	PS-N	PS-S	L2	L3
Total Precipitable Water	kg/m2	0.01	EQR	-	-	○	○
Cloud Liquid Water	kg/m2	0.001	EQR	-	-	○	○
Precipitation	mm/h	0.01	EQR	-	-	○	○
Sea Surface Temperature	C	0.01	EQR	-	-	○	○
Sea Surface Wind speed	m/s	0.01	EQR	-	-	○	○
Sea Ice Concentration	%	0.1	-	SIC	SIC	○	○
Snow Depth	cm	0.1	EQR	SND	-	○	○
Soil Moisture Content	%	0.1	EQR	-	-	○	○
Brightness Temperature (6GHz)	constant	constant	EQR	SIC	SIC	-	○
Brightness Temperature (7GHz)	constant	constant	EQR	SIC	SIC	-	○
Brightness Temperature (10GHz)	constant	constant	EQR	SIC	SIC	-	○
Brightness Temperature (18GHz)	constant	constant	EQR	SIC	SIC	-	○
Brightness Temperature (23GHz)	constant	constant	EQR	SIC	SIC	-	○
Brightness Temperature (36GHz)	constant	constant	EQR	SIC	SIC	-	○
Brightness Temperature (89GHz)	constant	constant	EQR	SIC	SIC	-	○

3 Programming using AMTK

The flow that the AMSR2 data is read and written with C and Fortran is show n below by the use of AMTK. Please show the flow of the programming here, and C language refer to the sample program of Fortran an actual source.

3.1 Flow of programming

(1) New making of HDF file



The include file for AMTK is specified.

The variable used is defined.

The memory area of the written data is allocated

The HDF file is opened in the writing mode. When the HDF file doesn't exist, it is newly made.

The meta data is written in the HDF file.

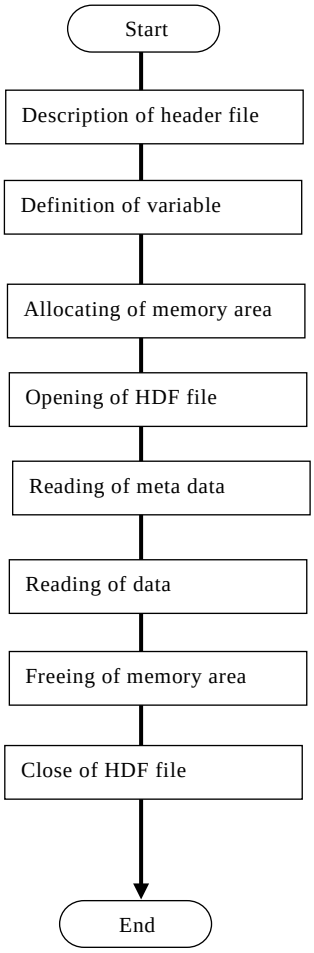
The data set is newly made for the HDF file.

Data is written in the HDF file.

The memory area is opened.

The HDF file is closed

(2) Reading of data from HDF file



The include file for AMTK is specified.

The variable used is defined.

The memory area of the read data is allocated.

The HDF file is opened in the reading mode.

The meta data is read from the HDF file.

Data is read to the memory area specifying the data set of the HDF file.

The memory area is opened.

The HDF file is closed.

3.2 C programming

3.2.1 C sample programming

It is Table 3.2 as for the sample program of C language It shows in one.

Table 3.2-1 sample programming

File name	Description of sample	Remark
sample1.c	The two meta data sets and the navigation are made, and is read again.	It explains the program in Chapter 3.2.2. The executable file name is sample1
sample2_make_L1Bproduct.c	The meta data and the data set of L1B are made.	The executable file name is sample2
sample3_make_L2Lproduct.c	The meta data and the data set of L2 low resolution are made.	The executable file name is sample3
sample4_make_L2Hproduct.c	The meta data and the data set of L2 high resolution are made.	The executable file name is sample4
sample5_make_L3Dproduct.c	The meta data and the data set of L3 daily are made.	The executable file name is sample5
sample6_make_L3Mproduct.c	The meta data and the data set of L3 monthly are made.	The executable file name is sample6
sample7_ read_L1Bproduct.c	The meta data and the data set of L1B are made.	The executable file name is sample7
sample8_read_L2Lproduct.c	The meta data and the data set of L2 low resolution are output to the screen.	The executable file name is sample8
sample9_read_L2Hproduct.c	The meta data and the data set of L2 high resolution are output to the screen.	The executable file name is sample9
sample10_read_L3Dproduct.c	The meta data and the data set of L3 daily are output to the screen.	The executable file name is sample10
sample11_read_L3Mproduct.c	The meta data and the data set of L3 monthly are output to the screen.	The executable file name is sample11
sample12_make_L1Rproduct.c	The meta data and the data set of L1R are made.	The executable file name is sample12
sample13_read_L1Rproduct.c	The meta data and the data set of L1R are made.	The executable file name is sample13
sample_common.c	Common functions source for sample programs	-
sample_common.h	Common functions header for sample programs	-

Please refer to README.txt of the directory with the sample program for details.

3.2.2 Description of programming

It explains the meta data of sample1.c, and here, the data set of 2 and the navigation is made, and it explains the part that relates to HDF about the read program.

(1) New making of HDF file

(a) Description of header file

#include "AMTK.h"	Header file in AMTK library

(b) Definition of variable

hid_t file_id;	Header file in AMTK library
float *navi = NULL;	The address of the navigation data is defined. The address of the navigation data is defined.
int size[3];	The variable that specifies the size of the data set is defined.

(c) Allocating of memory area

navi = (float *)malloc(sizeof(float) * scan_size * 6);	The memory area of the navigation data is allocated.

(d) Opening of HDF file

file_id = AMTK_openH5_Write(FN_SAMPLE_PRODUCT_L1B, AM2_CREATE_MODE);	The HDF file is opened.

(e) Writing of meta data

AMTK_setMetaData(file_id,metadata_name[i], metadata_value[i]);	The meta data is written.

(f) Making of data set

size[0] = scan_size;
size[1] = 0;
size[2] = 0;
	The size of the data set of the navigation data is specified.

AMTK_setDimSize(file_id, AM2_NAVI, size);	The data set of the navigation data is made.

(g) Writing of data

AMTK_set_SwathFloat(file_id, navi, scan_start, scan_end, AM2_NAVI);	Data is written in the data set.

(h) Free of memory area

free(navi);	The memory area is freed

(i) Close of HDF file

AMTK_closeH5_Write(file_id);	The HDF file is closed.

(2) Reading of data from HDF file

(a) Description of header file

#include "AMTK.h"	Header file in AMTK library

(b) Definition of variable

hid_t file_id;	Header file in AMTK library
float *navi = NULL;	The address of the navigation data is defined.
int size[3];	The variable that specifies the size of the data set is defined.
int ret;	It is a variable that stores the return value.

(c) Allocating memory area

ret = AMTK_getDimSize(file_id, AM2_LATLON_LO, &size[0]);	The size of the memory area of the necessity in the navigation data is acquired.
navi = (float *)malloc(sizeof(float) * scan_size * 6);	The memory area of the navigation data is allocated.

(d) Opening of HDF file

file_id = AMTK_openH5(FN_SAMPLE_PRODUCT_L1B);	The HDF file is opened.

(e) Reading of meta data

AMTK_getMetaData(file_id, META_PRODUCTNAME_NAME, metadata_&value);	The meta data is read.

(f) Reading of data

AMTK_get_SwathFloat(file_id, &navi, scan_start, scan_end, AM2_NAVI);	Data is read from the data set.

(g) Freeing of memory area

free(navi);	The memory area is freed.

(h) Close of HDF file

AMTK_closeH5(file_id);	The HDF file is closed.

3.2.3 C sample (sample1.c) program

The whole of sample1.c explains by 3.2.2 is shown below.

```
#include <stdio.h>
#include <stdlib.h>
#include "AMTK.h"

#define FN_SAMPLE_PRODUCT_L1B          "../data/sample1_IOToolKit.h5"
#define FN_SAMPLE_META_DATA_NAME_SN    "ShortName"
#define FN_SAMPLE_META_DATA_SN         "AMSR2-L1B"

#define FN_SAMPLE_META_DATA_NAME_OV    "OverlapScans"
#define FN_SAMPLE_META_DATA_OV         "20"

#define SCENE_SCAN_NUM    2000     /* */
#define OVERLAP_SCAN_NUM    20     /* */

/** @file sample1.c
 * I/O toolkit sample program main
 *
 * @author       NEC
 * @date         2010/07/31
 */
int main(int argc, char *argv[]){
     hid_t file_id;
     float *navi = NULL;
     int size[3];
     int scan_size = SCENE_SCAN_NUM + OVERLAP_SCAN_NUM * 2;
     int scan_start = 1 - OVERLAP_SCAN_NUM;
     int scan_end = SCENE_SCAN_NUM + OVERLAP_SCAN_NUM;
     char  granuleID[50];
        char  *value;
     int i;

     /* Newly make HDF Dataset  */
     navi = (float *)malloc(sizeof(float) * scan_size * 6);

     file_id = AMTK_openH5_Write(FN_SAMPLE_PRODUCT_L1B, AM2_CREATE_MODE );

     AMTK_setMetaDataName(file_id, FN_SAMPLE_META_DATA_NAME_SN, FN_SAMPLE_META_DATA_SN);
     AMTK_setMetaDataName(file_id, FN_SAMPLE_META_DATA_NAME_OV, FN_SAMPLE_META_DATA_OV);

     size[0] = scan_size;
     size[1] = 0;
     size[2] = 0;

     AMTK_setDimSize(file_id, AM2_NAVI, size);

      for( i=scan_start ; i< scan_end ; i++){
          navi[i+OVERLAP_SCAN_NUM-1] = (float)i*0.1;
     }
     AMTK_set_SwathFloat(file_id, navi, scan_start,
                                        scan_end, AM2_NAVI);

     free( navi );

     AMTK_closeH5_Write(file_id);

     /*    Read HDF Dataset  */
     navi = (float *)malloc(sizeof(float) * scan_size * 6);

     file_id = AMTK_openH5(FN_SAMPLE_PRODUCT_L1B);

        value = &granuleID[0];
     AMTK_getMetaDataName(file_id,"ShortName",&value) ;

     AMTK_get_SwathFloat(file_id, &navi, scan_start,
                                        scan_end, AM2_NAVI);

     printf( "ShortName=%s navi[0]=%f ¥n", granuleID, navi[OVERLAP_SCAN_NUM]);
```

```
        free( navi );

        AMTK_closeH5(file_id);

        return 0;
}
```

3.2.4 Compile and executions

Make the pattern file (named Makefile) by typing the following command.

```
$ ./configure
```

The file related to HDF is retrieved by the automatic operation as well as inst alling AMTK and Makefile is generated. Please correct the definition of Makefile generated with the configure command in the editor when it wants to change the compiler, and you want to edit passing the file related to HDF. If you installed HDF5 library to arbitrary directory, you can set the library with [--with-hdf-lib], [--with-hdf-include] options.

- Compilation command

```
CC= Compilation command of use compiler
CFLAGS= Option of use compiler
```

- HDF library

```
HDFLIB=/ Directory where HDF library exists
```

- HDF Directory

```
HDFINC=/ HDF Directory where include file exists
```

The make command to make an execute file is executed.

```
$ make
```

When the make command is executed, the execute form of all sample program is made for a present directory. Each sample program is executed. The example of ex

ecuting sample1 is shown as follows.

```
$./sample1
ShortName=AMSR2-L1B  navi[0]=0.1000000
```

"ShortName=AMSR2-L1B navi 0=0.1000000" message is output when sample1 is ex ecuted, and the HDF file named sample1_IOToolKit.h5 is made for the data direct ory of the sample directory.

Other samples are similar the compilation and execution.

3.3 Fortran Programing

As for Fortran, two kinds of types for f95 and for f77 are prepared. There is f95 type in the Fortran directory and there is f77 type in the Fortran77 directory. The file composition and both execution methods are the same. Here, it explains Fortran (for f95).

3.3.1 Fortran sample program

The table shows the sample program of Fortran.

Table 3.3-1 Fortran sample program

File name	Description of sample	Remark
sample1.f	The two meta data sets and the navigation are made, and is read again.	It explains the program in Chapter 3.3.2. The executable file name is sample1
sample2_make_L1Bproduct.f	The meta data and the data set of L1B are made.	The executable file name is sample2
sample3_make_L2Lproduct.f	The meta data and the data set of L2 low resolution are made.	The executable file name is sample3
sample4_make_L2Hproduct.f	The meta data and the data set of L2 high resolution are made.	The executable file name is sample4
sample5_make_L3Dproduct.f	The meta data and the data set of L3 daily are made.	The executable file name is sample5
sample6_make_L3Mproduct.f	The meta data and the data set of L3 monthly are made.	The executable file name is sample6
sample7_ read_L1Bproduct.f	The meta data and the data set of L1B are made.	The executable file name is sample7
sample8_read_L2Lproduct.f	The meta data and the data set of L2 low resolution are output to the screen.	The executable file name is sample8
sample9_read_L2Hproduct.f	The meta data and the data set of L2 high resolution are output to the screen.	The executable file name is sample9
sample10_read_L3Dproduct.f	The meta data and the data set of L3 daily are output to the screen.	The executable file name is sample10
sample11_read_L3Mproduct.f	The meta data and the data set of L3 monthly are output to the screen.	The executable file name is sample11
sample12_make_L1Rproduct.c	The meta data and the data set of L1R are made.	The executable file name is sample12
sample13_read_L1Rproduct.c	The meta data and the data set of L1R are made.	The executable file name is sample13
sample_common.c	Common functions source for sample programs	-
sample_common.h	Common functions header for sample programs	-

Please refer to README.txt of the directory with the sample program for details.

3.3.2 Description of programming

In here, it explains the part that relates to HDF in the program that makes two meta data of sample1.f and navigation data set and read it.

(1) New making of HDF file

(a) Description of header file

include 'AMTK_f.h'	Header file in AMTK library

(b) Definition of variable

Integer file_id	Header file in AMTK library
real navi(2000)	The address of the navigation data is defined.
Integer size(3)	The variable that specifies the size of the data set is defined.

(c) Opening of HDF file

file_id = AMTK_openH5_Write(fname, AM2_CREATE_MODE)	The HDF file is opened.

(d) Writing of meta data

status = AMTK_setMetaDataName(file_id, shortname_ID,shortname)	The meta data is written.
status = AMTK_setMetaDataName(file_id, OverlapScans_ID, * OverlapScans)	

(e) Making of data set

size(1) = 2040 size(2) = 0 size(3) = 0	The size of the data set of the navigation data is specified.
status = AMTK_setDimSize(file_id,AM2_NAVI, size)	The data set of the navigation data is made.

(f) Writing of data

status = AMTK_set_SwathFloat(file_id,navi,-19,2020, AM2_NAVI)	Data is written in the data set.

(g) Close of HDF file

status = AMTK_closeH5_Write(file_id)	The HDF file is closed.

(2) Reading of data from HDF file

(a) Opening of HDF file

file_id = AMTK_openH5(fname);	The HDF file is opened.

(b) Reading of meta data

status = AMTK_getMetaDataName(file_id,shortname_ID,granuleID)	The meta data is written.

(c) Reading of data

status = AMTK_get_SwathFloat(file_id, navi,1,2000, AM2_NAVI)	Data is read from the data set.

(d) Close of HDF file

status = AMTK_closeH5(file_id)	The HDF file is closed.

3.3.3 Fortran(sample1_f) Sample program

```
      Program main

       include 'AMTK_f.h'

C       Implicit NONE
       character*30 fname
       data fname/'../data/sample1_f_IOToolKit.h5'/
       Integer file_id
       Integer status
       integer*1 char1
       integer*2 char2
       integer*1 char3(10)
       character*9 shortname_ID
       data shortname_ID/'ShortName'/
       character*9 shortname
       data shortname/'AMSR2-L1B'/
       character*12 OverlapScans_ID
       data OverlapScans_ID/'OverlapScans'/
       character*2 OverlapScans
       data OverlapScans/'20'/
       character*40 granuleID
       Integer size(3)
       Integer i

       real navi(2000)

       file_id = AMTK_openH5_Write( fname , AM2_CREATE_MODE )

       status = AMTK_setMetaDataName(file_id, shortname_ID,shortname)

       status = AMTK_setMetaDataName(file_id, OverlapScans_ID,
      * OverlapScans)

       size(1) = 2040
       size(2) = 0
       size(3) = 0

       status = AMTK_setDimSize(file_id,AM2_NAVI, size)

       i = 1
       Do i=1,2000
            navi(i) = i*0.1
       End do

       status = AMTK_set_SwathFloat(file_id,navi,-19,2020, AM2_NAVI)

       status = AMTK_closeH5_Write(file_id)

       file_id = AMTK_openH5( fname );

       status = AMTK_getMetaDataName(file_id,shortname_ID,granuleID)

       status = AMTK_get_SwathFloat(file_id, navi,-19,2020, AM2_NAVI)

       status = AMTK_closeH5(file_id);

       Write(*,*) 'ShortName=', shortname, ' navi(1)=', navi(1)

       Stop
       End
```

3.3.4 Compilations and executions

Makefile for the make command execution is made first.

```
$ ./configure
```

The make command of the execute form making is executed. Please correct the definition of Makefile generated with the configure command in the editorwhen it wants to change the compiler, and you want to edit passing the file related to HDF.

If you installed HDF5 library to arbitrary directory, you can set the library wit h [--with-hdf-lib], [--with-hdf-include] options.

In the case of SGI environment, the Makefile must be added "-64" at the FFLAGS.

```
FFLAGS = ...(*snip*)... -64          In the case of SGI environment
```

○HDF library file

```
--with-hdf-lib=/HDF5/shread/lib
```

○HDF include file

```
--with-hdf-include=/HDF5/shread/include
```

- Compilation command

```
CC= Compilation command of use compiler
CFLAGS= Option of use compiler
```

- HDF library

```
HDFLIB=/ Directory where HDF library exists
```

- HDF directory

```
HDFINC=/ Directory where HDF include file exists
```

```
$ make
```

When the make command is executed, the execute form of all sample program is

made for a present directory. Each sample program is executed. The example of ex ecuting sample1 is shown as follows.

```
$./sample1
ShortName=AMSR2-L1B  navi[0]=0.1000000
```

"ShortName=AMSR2-L1B navi 0=0.1000000" message is output when sample1 is ex ecuted, and the HDF file named sample1_IOToolKit.h5 is made for the data direct ory of the sample directory.

Other samples are similar the compilation and execution.

4 Function composition

Function composition of AMTK is shown in Fig3.3-1.

The offered function is divided by the input function and the output function.

1） Input function

(1)Level 1 product(L1A ,L1B ,L1R)input function

(2)Level 2 product(L2)input function

(3)Level 3 product(L3)input function

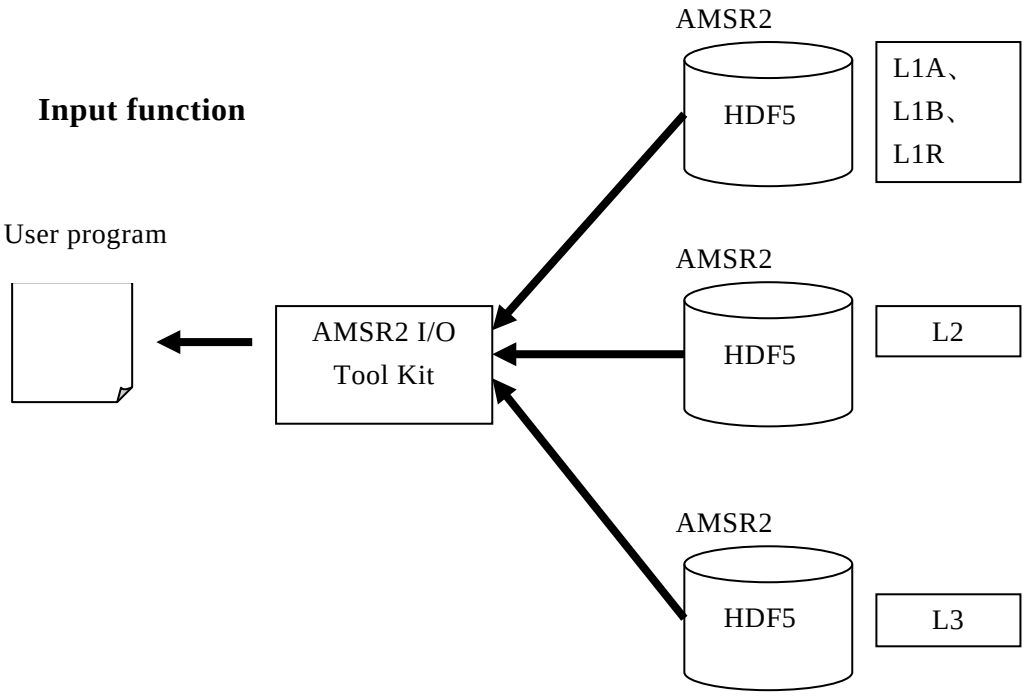


Fig 4-1 AMTK product input function

2） Output function

(1)Level 1 product(L1A ,L1B ,L1R)output function

(2)Level 2 product (L2)output function

(3)Level 3 product (L3)output function

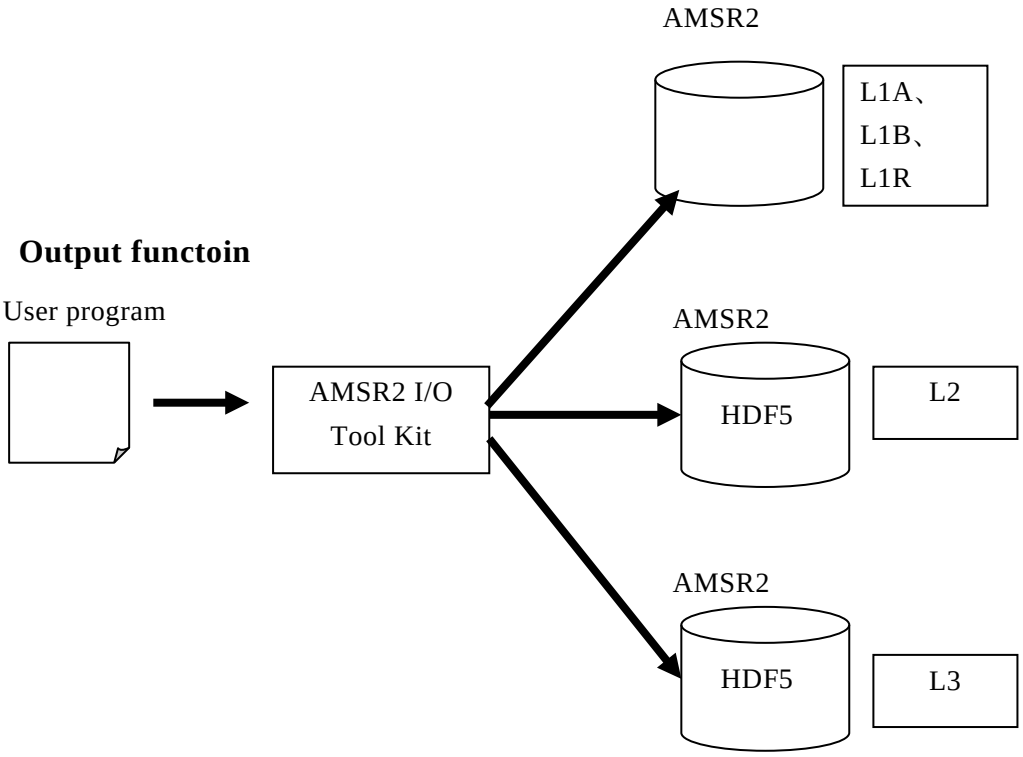


Fig.4-2 AMTK product output function

5 API of function

5.1 C- Language

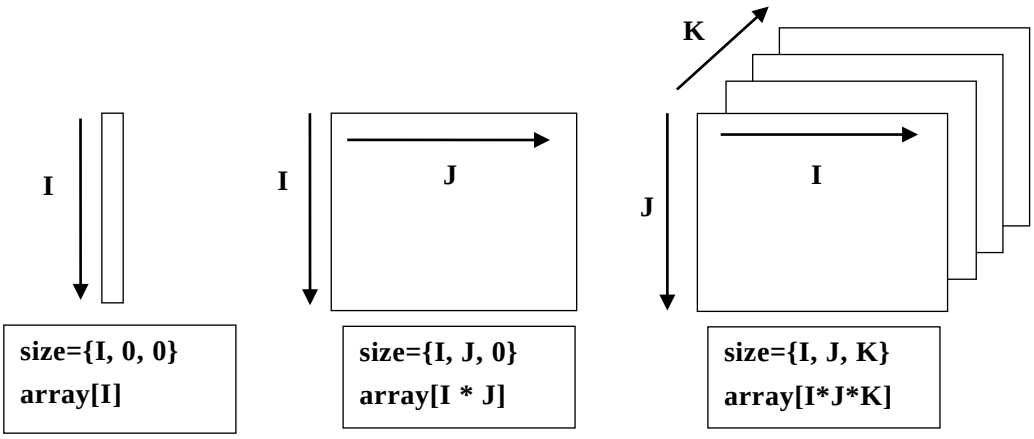
5.1.1 Common function

(1) L1A ,L1B ,L1R ,L2 ,L3 input functions

Product file opening (reading only)				
The HDF product file is opened with read only.				
hid_t AMTK_openH5(char *file_name)				
Name	Type	IN/OUT	Size	Description
return value				
HDF_file_id	hid_t	output	1	Normal：HDF access file id is returned Error：A negative value returns when it is not possible to acquire it.
parameter				
file_name	char *	input	1	AMSR2 HDFfile name

Product file close				
The HDF product file is closed.				
int AMTK_closeH5(hid_t HDF_file_id)				
Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal：0 Error：A negative value returns.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id

Acquisition of data dimension size				
The data dimension size is acquired from product. — When the access label intended for plural Datasets is specified, the error is returned.				
int AMTK_getDimSize(hid_t HDF_file_id, int access_lbl , int *size)				
Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal：The array number of dimension returns. Error：A negative value returns when it is not possible to acquire it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
access_lbl	int	input	1	Please refer to "6.1.1 the HDF access label".
size	int *	output	1	Dimension Size (Please refer the fig below)



One-dimensional data

Two-dimensional data

Three-dimensional data

Refer to the 6.1.2 section.

Acquisition of meta data(meta data name)				
The meta data is acquired by specifying the meta data name of the product. — When the memory area where the meta data value is stored is NULL, the error is returned.				
int AMTK_getMetaDataName(hid_t HDF_file_id, char *name, char **value)				
Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal : The number of meta data character returns. Error : A negative value returns when it is not possible to acquire it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
name	char *	input	1	meta data name
value	char **	output	1	meta data value

Acquisition of meta data(index value)				
The meta data is acquired by specifying the index value of the product. — When the memory area where the meta data value is stored is NULL, the error is returned.				
int AMTK_getMetaData(hid_t HDF_file_id, int index, char **value)				
Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal : The number of meta data character returns. Error : A negative value returns when it is not possible to acquire it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
index	int	input	1	meta data index value
value	char **	output	1	meta data value

Acquisition of scanning time				
The scanning time is acquired. - The scan number appoints a start scan number and an end scan number. When an end scan number is smaller than a start scan number, an error returns. When an end scan number is bigger than a scan number stored away by appointed HDF data set, It acquires data stored away by HDF data set. - The user using a function allocate memory area storing data. A data size is “sizeof(AM2_COMMON_SCANTIME)*dimension size(AMTK_getDimSize function reference)” Ex.) Scan Time (scan): AM2_COMMON_SCANTIME *p_data = malloc(sizeof(AM2_COMMON_SCANTIME) * scan)				
int AMTK_getScanTime(hid_t HDF_file_id, int from_scan, int to_scan, AM2_COMMON_SCANTIME **scan_time)				
Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal : The number of acquisition scannings returns. Error : A negative value returns when it is not possible to acquire it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
from_scan	int	input	1	Scan number of acquisition start
to_scan	int	input	1	Scan number of acquisition end
scan_time	AM2_COMMON_SCANTIME **	output	1	Data structure

Acquisition of Latitude longitude data

The Lat/Long data of the range of an appointed scan number is acquired.

- The scan number appoints a start scan number and an end scan number.
  When an end scan number is smaller than a start scan number, an error returns.
  When an end scan number is bigger than a scan number stored away by appointed HDF data set, It acquires data stored away by HDF data set.
- The user using a function allocate memory area storing data.
  A data size is "sizeof(AM2_COMMON_ LATLON)*dimension size(AMTK_getDimSize function reference)"

Ex.) Latitude of Observation Point for 89A/89B (scan x 486)
: AM2_COMMON_LATLON *p_data = malloc(sizeof(AM2_COMMON_LATLON) * scan * 486)

int AMTK_getLatLon(hid_t HDF_file_id, AM2_COMMON_LATLON **latitudelongitude, int from_scan, int to_scan, int access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal：The number of acquired scan is returned. Error：A negative value returns when it is not possible to acquire it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
latitudelongitude	AM2_COMMON_LATLON **	output	1 (x N scan)	Data structure
from_scan	int	input	1	Scan number of acquisition start
To_scan	int	input	1	Scan number of acquisition end
access_lbl	int	input	1	Please refer to "6.1.1.6 the LATLON function".

Acquisition of L1 quality information

The quality information of L1 data is acquired.

- The scan number appoints a start scan number and an end scan number.
  When an end scan number is smaller than a start scan number, an error returns.
  When an end scan number is bigger than a scan number stored away by appointed HDF data set, It acquires data stored away by HDF data set.
- The user using a function allocate memory area storing data.
  A data size is "sizeof(AM2_COMMON_ LATLON)*dimension size(AMTK_getDimSize function reference)"

Ex.) Scan Data Quality (scan x 512) # data set is char type. x sizeof(AMTK_SCAN_DATA_QUALITY)size
: AMTK_SCAN_DATA_QUALITY *p_data = malloc(sizeof(AMTK_SCAN_DATA_QUALITY) * scan)

int AMTK_getScanDataQuality(hid_t HDF_file_id, AMTK_SCAN_DATA_QUALITY **quality, int from_scan, int to_scan, int access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal：The number of acquired scan is returned. Error：A negative value returns when it is not possible to acquire it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
quality	AMTK_SCAN_DATA_QUALITY **	output	1	Data structure
from_scan	int	input	1	Scan number of acquisition start
to_scan	int	input	1	Scan number of acquisition end
access_lbl	int	input	1	Please refer to "6.1.1 the HDF access label".

Acquisition of Integer Type data

The data of integer Type is acquired specifying the HDF access label.

- The data of the correspondence is acquired specifying the scanning number.
- The specification of the scanning number specifies the beginning scanning number and the end scanning number.
  The error returns when the end scanning number is smaller than the beginning scanning number.
  When the end scanning number is larger than the scanning number accommodated by HDF Dataset, the data accommodated by HDF Dataset is acquired.
- The user who uses the function has to alocate the memory area where data is stored.
  Size specifies "Number of sizeof(int) * line number * scannings".
- When the memory area specification that stores data is NULL, the memory area is secured in the function.

int AMTK_get_SwathInt(hid_t HDF_file_id, int **data, int from_scan, int to_scan, int access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal：The number of acquired scan is returned. Error：A negative value returns when it is not possible to acquire it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	int **	output	1	Memory area where data is stored
from_scan	int	input	1	Scan number of acquisition start
to_scan	int	input	1	Scan number of acquisition end
access_lbl	int	input	1	Please refer to "6.1.1 the HDF access label".

Acquisition of floating Type data

The data of floating Type is acquired specifying the HDF access label.

- The data of the correspondence is acquired specifying the scanning number.
- The specification of the scanning number specifies the beginning scanning number and the end scanning number.
  The error returns when the end scanning number is smaller than the beginning scanning number.
  When the end scanning number is larger than the scanning number accommodated by HDF Dataset, the data accommodated by HDF Dataset is acquired.
- The user who uses the function has to alocate the memory area where data is stored.
  Size specifies "Number of sizeof(float) * line number * scannings".
- When the memory area specification that stores data is NULL, the memory area is secured in the function.

int AMTK_get_SwathFloat(hid_t HDF_file_id, float **data, int from_scan, int to_scan, int access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal：The number of acquired scan is returned. Error：A negative value returns when it is not possible to acquire it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	float **	output	1	Memory area where data is stored
from_scan	int	input	1	Scan number of acquisition start
to_scan	int	input	1	Scan number of acquisition end
access_lbl	int	input	1	Please refer to "6.1.1 the HDF access label".

Acquisition of double Type data

The data of double Type is acquired specifying the HDF access label.

- The data of the correspondence is acquired specifying the scanning number.
- The specification of the scanning number specifies the beginning scanning number and the end scanning number.
  The error returns when the end scanning number is smaller than the beginning scanning number.
  When the end scanning number is larger than the scanning number accommodated by HDF Dataset, the data accommodated by HDF Dataset is acquired.
- The user who uses the function has to alocate the memory area where data is stored.
  Size specifies "Number of sizeof(double) * line number * scannings".
- When the memory area specification that stores data is NULL, the memory area is secured in the function.

int AMTK_get_SwathDouble(hid_t HDF_file_id, double **data, int from_scan, int to_scan, int access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal：The number of acquired scan is returned. Error：A negative value returns when it is not possible to acquire it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	double **	output	1	Memory area where data is stored
from_scan	int	input	1	Scan number of acquisition start
to_scan	int	input	1	Scan number of acquisition end
access_lbl	int	input	1	Please refer to "6.1.1 the HDF access label".

Acquisition of unsigned CharType data

The data of unsigned CharType is acquired specifying the HDF access label.

- The data of the correspondence is acquired specifying the scanning number.
- The specification of the scanning number specifies the beginning scanning number and the end scanning number.
  The error returns when the end scanning number is smaller than the beginning scanning number.
  When the end scanning number is larger than the scanning number accommodated by HDF Dataset, the data accommodated by HDF Dataset is acquired.
- The user who uses the function has to alocate the memory area where data is stored.
  Size specifies "Number of sizeof(unsigned char) * line number * scannings".
- When the memory area specification that stores data is NULL, the memory area is secured in the function.

int AMTK_get_SwathUChar(hid_t HDF_file_id, unsigned char **data, int from_scan, int to_scan, int access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal：The number of acquired scan is returned. Error：A negative value returns when it is not possible to acquire it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	unsigned char **	output	1	Memory area where data is stored
from_scan	int	input	1	Scan number of acquisition start
to_scan	int	input	1	Scan number of acquisition end
access_lbl	int	input	1	Please refer to "6.1.1 the HDF access label".

(2) L1A ,L1B ,L1R ,L2 ,L3 Output functions

Product file opening (writing only)				
The HDF product file is opened with write only.				
hid_t AMTK_openH5_Write(char *file_name, int mode)				
Name	Type	IN/OUT	Size	Description
return value				
HDF_file_id	hid_t	output	1	Normal：HDF access file id Error：A negative value returns when it is not possible to open.
parameter				
file_name	char *	input	1	AMSR2 HDFfile name
mode	int	input	1	File open mode AM2_CREATE_MODE : New making The existing file is deleted. AM2_RW_MODE : Superscription It becomes an error when there is no existing file.

Product file close				
The HDF product file is closed.				
int AMTK_closeH5_Write(hid_t HDF_file_id)				
Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal：0 Error：A negative value returns.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id

HDF Dataset making				
HDF Dataset of data dimention size is made. — When generating it, HDF Dataset is initialized by the missing value.				
int AMTK_setDimSize(hid_t HDF_file_id,int access_lbl , int *size)				
Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal：0 Error：A negative value returns when it is not possible to set it
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
access_lbl	int	input	1	Please refer to "6.1.1 the HDF access label".
size	int *	input	1	Dimention size (For detailed information, three dimention array. Please refer to function Description of AMTK_getDimSize.)

Meta data setting				
The meta data is set by the meta data name of the HDF product.				
int AMTK_setMetaDataName(hid_t HDF_file_id, char *name, char *value)				
Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal：0 Error： A negative value returns when it is not possible to set it
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
name	char *	input	1	Meta data name
value	char *	input	1	Meta data value

Setting at scanning time

Scanning time data (Year, Month, Day, Hour, Minute, second, mili second) is converted to TAI93 real integer and written in the specified scan number.

- The scan number appoints a start scan number and an end scan number.

When an end scan number is smaller than a start scan number, an error returns.

When an end scan number is bigger than a scan number stored away by appointed HDF data set, It acquires data stored away by HDF data set.

- The user using a function allocate memory area storing data.

A data size is "sizeof(AM2_COMMON_SCANTIME)*dimension size(AMTK_getDimSize function reference)"

Ex.) Scan Time (scan): AM2_COMMON_SCANTIME *p_data = malloc(sizeof(AM2_COMMON_SCANTIME) * scan)

int AMTK_setScanTime(hid_t HDF_file_id, AM2_COMMON_SCANTIME *scan_time, int from_scan, int to_scan)

Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal:0 Error:A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
scan_time	AM2_COMMON_SCANTIME *	input	1	Data structure
from_scan	int	input	1	beginning scanning number
to_scan	int	input	1	end scanning number

Latitude longitude data setting

The Lat/Long data is written in the specified scan number.

- The scan number appoints a start scan number and an end scan number.

When an end scan number is smaller than a start scan number, an error returns.

When an end scan number is bigger than a scan number stored away by appointed HDF data set, It acquires data stored away by HDF data set.

- The user using a function allocate memory area storing data.

A data size is "sizeof(AM2_COMMON_ LATLON)*dimension size(AMTK_getDimSize function reference)"

Ex) Latitude of Observation Point for 89A/89B (scan x 486)

: AM2_COMMON_LATLON *p_data = malloc(sizeof(AM2_COMMON_LATLON) * scan * 486)

int AMTK_setLatLon(hid_t HDF_file_id, AM2_COMMON_LATLON *latitudelongitude, int from_scan, int to_scan, int access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal:0 Error:A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
latitudelongitude	AM2_COMMON_LATLON *	input	1 (x N scan)	Data structure
from_scan	int	input	1	beginning scanning number
to_scan	int	input	1	end scanning number
access_lbl	int	input	1	Please refer to "6.1.1.6 the LATLON function".

Quality information setting

The qualitiy information of L1 data is set.
- The scan number appoints a start scan number and an end scan number.
When an end scan number is smaller than a start scan number, an error returns.
When an end scan number is bigger than a scan number stored away by appointed HDF data set, It acquires data stored away by HDF data set.
- The user using a function allocate memory area storing data.
A data size is “sizeof(AM2_COMMON_SCANTIME)*dimension size(AMTK_getDimSize function reference)”
Ex)) Scan Data Quality (scan x 512) # data set is char type. x sizeof(AMTK_SCAN_DATA_QUALITY)size
: AMTK_SCAN_DATA_QUALITY *p_data = malloc(sizeof(AMTK_SCAN_DATA_QUALITY) * scan)

int AMTK_set_ScanDataQuality(hid_t HDF_file_id, AMTK_SCAN_DATA_QUALITY *quality, int from_scan, int to_scan, int access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal:0 Error:A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
quality	AMTK_SCAN_DATA_QUALITY *	output	1	Data structure
from_scan	int	input	1	beginning scanning number
to_scan	int	input	1	end scanning number
access_lbl	int	input	1	Please refer to "6.1.1 the HDF access label".

Integer Type data setting

Data is set to HDF Dataset as data of integer Type by specifying the HDF access label.
- The data of the correspondence is acquired specifying the scanning number.
- The specification of the scanning number specifies the beginning scanning number and the end scanning number.
The error returns when the end scanning number is smaller than the beginning scanning number.
When the end scanning number is larger than the scanning number accommodated by HDF Dataset, the data accommodated by HDF Dataset is acquired.
- The user who uses the function has to alocate the memory area where data is stored.
Size specifies "Number of sizeof(int) * line number * scannings".
- When the numerical value that cannot be accommodated in HDF Dataset exists in the setting data, correspondence data is replaced with the error value and it writes it in HDF. And a negative value of the warning is set to return value.

int AMTK_set_SwathInt(hid_t HDF_file_id, int *data, int from_scan, int to_scan, int access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal:0 Error:A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	int *	input	1	Memory area where data is stored
from_scan	int	input	1	beginning scanning number
to_scan	int	input	1	end scanning number
access_lbl	int	input	1	Please refer to "6.1.1 the HDF access label".

Floating Type data setting

Data is set to HDF Dataset as data of floating Type by specifying the HDF access label.

- The data of the correspondence is acquired specifying the scanning number.
- The specification of the scanning number specifies the beginning scanning number and the end scanning number.
  The error returns when the end scanning number is smaller than the beginning scanning number.
  When the end scanning number is larger than the scanning number accommodated by HDF Dataset, the data accommodated by HDF Dataset is acquired.
- The user who uses the function has to alocate the memory area where data is stored.
  Size specifies "Number of sizeof(float) * line number * scannings".
- When you convert the value of the floating number into the integral value by the scale factor of HDF Dataset of the correspondence, the first place of the decimal point is rounded off after it divides in the scale factor.
  Ex.) Navigation Data (scan x 6): float *p_data = malloc(sizeof(float) * scan * 6)

int AMTK_set_SwathFloat(hid_t HDF_file_id, float *data, int from_scan, int to_scan, int access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal:0 Error:A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	float *	input	1	Memory area where data is stored
from_scan	int	input	1	beginning scanning number
to_scan	int	input	1	end scanning number
access_lbl	int	input	1	Please refer to "6.1.1 the HDF access label".

Double Type data setting

Data is set to HDF Dataset as data of double Type by specifying the HDF access label.

- The data of the correspondence is acquired specifying the scanning number.
- The specification of the scanning number specifies the beginning scanning number and the end scanning number.
  The error returns when the end scanning number is smaller than the beginning scanning number.
  When the end scanning number is larger than the scanning number accommodated by HDF Dataset, the data accommodated by HDF Dataset is acquired.
- The user who uses the function has to alocate the memory area where data is stored.
  Size specifies "Number of size of(double) * line number * scannings".
  Ex.) Position in Orbit (scan): double *p_data = malloc(sizeof(double) * scan)

int AMTK_set_SwathDouble(hid_t HDF_file_id, double *data, int from_scan, int to_scan, int access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal:0 Error:A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	double *	input	1	Memory area where data is stored
from_scan	int	input	1	beginning scanning number
to_scan	int	input	1	end scanning number
access_lbl	int	input	1	Please refer to "6.1.1 the HDF access label".

Unsigned Char Type data setting

Data is set to HDF Dataset as data of Unsigned Char Type by specifying the HDF access label.

- The data of the correspondence is acquired specifying the scanning number.
- The specification of the scanning number specifies the beginning scanning number and the end scanning number.
  The error returns when the end scanning number is smaller than the beginning scanning number.
  When the end scanning number is larger than the scanning number accommodated by HDF Dataset, the data accommodated by HDF Dataset is acquired.
- The user who uses the function has to alocate the memory area where data is stored.
  Size specifies "Number of size of(unsigned char) * line number * scannings".
- When the numerical value that cannot be accommodated in HDF Dataset exists in the setting data, correspondence data is replaced with the error value and it writes it in HDF. And a negative value of the warning is set to return value.

Ex.) SPC Temperature Count (scan x 34): unsigned char *p_data = malloc(scan * 34)

int AMTK_set_SwathUChar(hid_t HDF_file_id, unsigned char *data, int from_scan, int to_scan, int access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal：0 Error：A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	unsigned char *	input	1	Memory area where data is stored
from_scan	int	input	1	beginning scanning number
to_scan	int	input	1	end scanning number
access_lbl	int	input	1	Please refer to "6.1.1 the HDF access label".

5.1.2 Input functions

(1) L1A, L1B ,L1R function

There is no input function peculiar to L1 because the common function is used.

(2) L2 input function

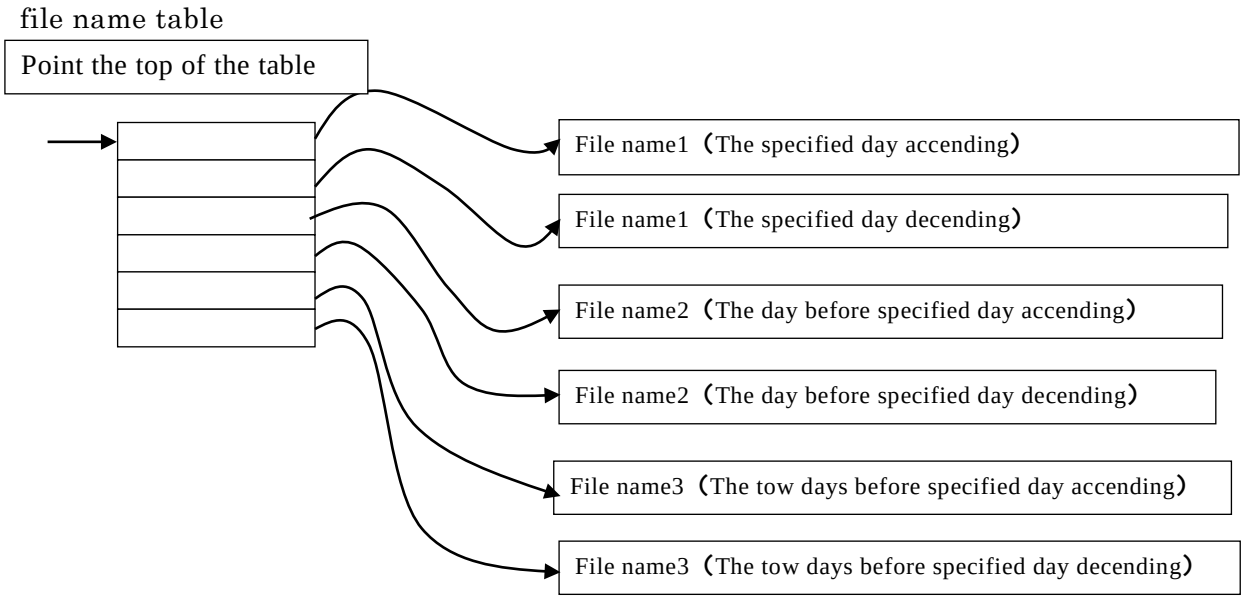
There is no input function peculiar to L2 because the common function is used.

(3) L3 input function

Acquistion of Integer Type data				
The data of integer Type is acquired specifying the HDF access label. - All of the array data of the L3 product specified with the access label are acquired. To get the information of array Size, it uses the AMTK_getDimSize function. - The user who uses the function secures Memory area where data is stored. Size specifies "sizeof(int)*Dimension". - When the Memory area where data is stored specification is NULL, the memory area is allocated in the function internally.				
int AMTK_get_GridInt(hid_t HDF_file_id, int **data, int access_lbl)				
Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal：The number of acquired scan is returned. Error：A negative value returns when it is not possible to acquire it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	int **	output	1	Memory area where data is stored
access_lbl	int	input	1	Please refer to "6.1.1 the HDF access label".

Acquistion of float type data **				
The data of float Type is acquired specifying the HDF access label. - All of the array data of the L3 product specified with the access label are acquired. To get the information of array Size, it uses the AMTK_getDimSize function. - The user who uses the function secures Memory area where data is stored. Size specifies "sizeof(int)*Dimension". - When the Memory area where data is stored specification is NULL, the memory area is allocated in the function internally.				
int AMTK_get_GridFloat(hid_t HDF_file_id, float **data, int access_lbl)				
Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal：The number of acquired scan is returned. Error：A negative value returns when it is not possible to acquire it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	float **	output	1	Memory area where data is stored
access_lbl	int	input	1	Please refer to "6.1.1 the HDF access label".

Average data acquisition on the third				
The average data of three days of level 3 is acquired. In the case the input file is not based on rules, the error number (-250 or more) is returned. Please refer to ”7 Error Number ID” for the details.				
int AMTK_get_Grid03D(float **average, char *name_table)				
Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal：0 Error：A negative value returns when it is not possible to acquire it.
parameter				
average	float **	output	M pixcel x N scan	Data structure 0.1-degree or 0.25-degree lattice is judged from a file name. The all specified files are from the same lattice.
name_table	char *	input	256	File name table (Please refer to the figure below).



5.1.3 Output functions

(1) L1 output function

There is no output function peculiar to L1 because the common function is used.

(2) L2 output function

There is no output function peculiar to L2 because the common function is used.

(3) L3 output function

Acquistion of Integer Type data				
The HDF Dataset data are set as integer type data in an appointed HDF access lavel. - All of the array data of the L3 product appointed with the access label are acquired. To get the information of array Size, it uses the AMTK_getDimSize function. - The user who uses the function secures Memory area where data is stored. Size specifies "size of(int)*Dimension". Ex.) Average Number (V) (Xpixel x Ypixel): int *p_data = malloc(sizeof(int) * Xpixel * Ypixel)				
int AMTK_get_GridInt(hid_t HDF_file_id, int **data, int access_lbl)				
Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal：0 Error：A negative value returns when it is not possible to acquire it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	int **	output	1	Memory area where data is stored
access_lbl	int	input	1	Please refer to "6.1.1 the HDF access label".

Acquistion of float Type data				
The HDF Dataset data are set as float type data in an appointed HDF access lavel. - All of the array data of the L3 product specified with the access label are acquired. To get the information of array Size, it uses the AMTK_getDimSize function. - The user who uses the function secures Memory area where data is stored. Size specifies "sizeof(int)*Dimension". Ex.) Standard Deviation (V) (Xpixel x Ypixel): float *p_data = malloc(sizeof(float) * Xpixel * Ypixel)				
int AMTK_get_GridInt(hid_t HDF_file_id, int **data, int access_lbl)				
Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal：0 Error：A negative value returns when it is not possible to acquire it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	int **	output	1	Memory area where data is stored
access_lbl	int	input	1	Please refer to "6.1.1 the HDF access label".

5.2 Fortran

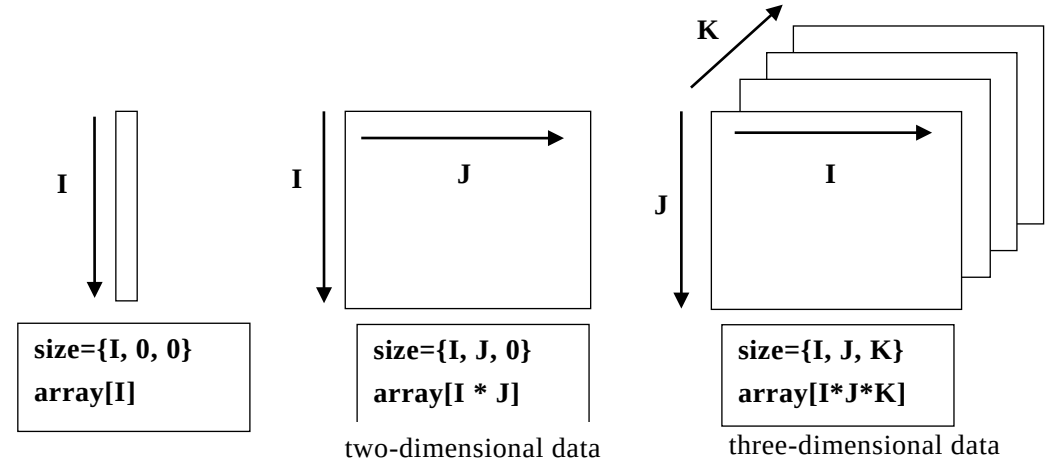
5.2.1 Common function

(1) L1A ,L1B ,L1R ,L2 ,L3 input functions

Product file opening (reading only)				
The HDF product file is opened with read only.				
hid_t = AMTK_openH5(file_name)				
Name	Type	IN/OUT	Size	Description
return value				
HDF_file_id	integer	output	1	Normal：HDF access file id is returned Error：A negative value returns when it is not possible to acquire it.
parameter				
file_name	character*500	input	1	AMSR2 HDFfile name

Product file close				
The HDF product file is closed.				
int AMTK_closeH5(hid_t HDF_file_id)				
Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal：0 Error：A negative value returns.
parameter				
HDF_file_id	integer	input	1	HDF access file id

Acquisition of the data dimension size				
The data dimension size is acquired from HDF product. — When the access label intended for plural Datasets is specified, the error is returned.				
status = AMTK_getDimSize(HDF_file_id, access_lbl , size)				
Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal：The array number of dimension returns. Error：A negative value returns when it is not possible to acquire it.
Parameter				
HDF_file_id	hid_t	input	1	HDF access file id
access_lbl	integer	input	1	Please refer to "6.1.1 the HDF access label".
size(3)	integer	output	1	Dimension Size (Please refer the fig below)



one-dimensional data

two-dimensional data

three-dimensional data

Refer to the 6.1.2 section.

Acquisition of meta data(meta data name)

The meta data is acquired by specifying the meta data name of the product.

— When the memory area where the meta data value is stored is NULL, the error is returned.

status = AMTK_getMetaDataName(HDF_file_id, name, value)

Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal : The number of meta data character returns. Error : A negative value returns when it is not possible to acquire it.
Parameter				
HDF_file_id	integer	input	1	HDF access file id
name	character*30	input	1	Meta data name
value	character*10	output	1	Meta data value

Acquisition of meta data(index value)

The meta data is acquired by specifying the index value of the product.

— When the memory area where the meta data value is stored is NULL, the error is returned.

status = AMTK_getMetaData(HDF_file_id, index, value)

Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal : The number of meta data character returns. Error : A negative value returns when it is not possible to acquire it.
Parameter				
HDF_file_id	integer	input	1	HDF access file id
index	integer	input	1	meta data index value
value	character*10	output	1	meta data value

Acquisition of scanning time

The scanning time is acquired.

- The scan number appoints a start scan number and an end scan number.
When an end scan number is smaller than a start scan number, an error returns.
When an end scan number is bigger than a scan number stored away by appointed HDF data set, It acquire data stored away by HDF data set.
- The user using a function allocate memory area storing data.
A data size is “sizeof(AM2_COMMON_SCANTIME)*dimension size(AMTK_getDimSize function reference)”
Ex.) Scan Time (scan): type(AM2_COMMON_SCANTIME) data(scan)

status = AMTK_getScanTime(HDF_file_id, from_scan, to_scan, scan_time)

Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal : The number of acquired scan is returned. Error : A negative value returns when it is not possible to acquire it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
from_scan	integer	input	1	Scan number of acquisition start
to_scan	integer	input	1	Scan number of acquisition end
scan_time	AM2_COMMON_SCANTIME	output	1	Data structure

Acquisition of Latitude longitude data

It takes out the Lat/Long data of the range of an appointed scan number from a HDF.

- The scan number appoints a start scan number and an end scan number.
When an end scan number is smaller than a start scan number, an error returns.
When an end scan number is bigger than a scan number stored away by appointed HDF data set, It acquires data stored away by HDF data set.
- The user using a function allocate memory area storing data.
A data size is "sizeof(AM2_COMMON_LATLON)*dimension size(AMTK_getDimSize function reference)"
Ex.) Latitude of Observation Point for 89A/89B (scan x 486) : type(AM2_COMMON_LATLON) data(486, scan)

status = AMTK_getLatLon(HDF_file_id, latitudelongitude, from_scan, to_scan, access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal : The number of acquired scan is returned. Error : A negative value returns when it is not possible to acquire it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
latitudelongitude	AM2_COMMON_LATLON *	output	1 (x N scan)	Data structure
from_scan	integer	input	1	Scan number of acquisition start
to_scan	integer	input	1	Scan number of acquisition end
access_lbl	integer	input	1	Please refer to "6.1.1.6 the LATLON function".

Acquisition of L1 quality information

The quality information of the L1 data is acquired.

- The scan number appoints a start scan number and an end scan number.
When an end scan number is smaller than a start scan number, an error returns.
When an end scan number is bigger than a scan number stored away by appointed HDF data set, It acquires data stored away by HDF data set.
- The user using a function allocate memory area storing data.
A data size is "512(AM2_COMMON_LATLON)*dimension size(AMTK_getDimSize function reference)"
Ex.) Scan Data Quality (scan x 512) # data set is char type. x sizeof(AMTK_SCAN_DATA_QUALITY)size
: type(AMTK_SCAN_DATA_QUALITY) data(scan)

int AMTK_getScanDataQuality(HDF_file_id, quality, from_scan, to_scan, access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	Int	output	1	Normal : The number of acquired scan is returned. Error : A negative value returns when it is not possible to acquire it.
Parameter				
HDF_file_id	hid_t	input	1	HDF access file id
quality	AMTK_SCAN_DATA_QUALITY *	output	1	Data structure
from_scan	Int	input	1	Scan number of acquisition start
to_scan	Int	input	1	Scan number of acquisition end
access_lbl	Int	input	1	Please refer to "6.1.1 the HDF access label".

Acquisition of Integer Type data

The data of integer Type is acquired specifying the HDF access label.

- The data of the correspondence is acquired specifying the scanning number.
- The specification of the scanning number specifies the beginning scanning number and the end scanning number.
  The error returns when the end scanning number is smaller than the beginning scanning number.
  When the end scanning number is larger than the scanning number accommodated by HDF Dataset, the data accommodated by HDF Dataset is acquired.
- The user who uses the function has to alocate the memory area where data is stored.
  Size specifies "Number of sizeof(int) * line number * scannings".
- When the memory area specification that stores data is NULL, the memory area is secured in the function.

Ex.) Hot Load Count 6 to 36 (12 x scan x 16): integer data(16 * scan * 12)

status = AMTK_get_SwathInt(HDF_file_id, data, from_scan, to_scan, access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	Integer	output	1	Normal：The number of scan acquired returns. Error：A negative value returns when it is not possible to acquire it.
Parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	Integer	output	1	Memory area where data is stored
from_scan	Integer	input	1	Scan number of acquisition start
to_scan	Integer	input	1	Scan number of acquisition end
access_lbl	Integer	input	1	Please refer to "6.1.1 the HDF access label"

Acquisition of floating Type data

The data of floating Type is acquired specifying the HDF access label.

- The data of the correspondence is acquired specifying the scanning number.
- The specification of the scanning number specifies the beginning scanning number and the end scanning number.
  The error returns when the end scanning number is smaller than the beginning scanning number.
  When the end scanning number is larger than the scanning number accommodated by HDF Dataset, the data accommodated by HDF Dataset is acquired.
- The user who uses the function has to alocate the memory area where data is stored.
  Size specifies "Number of size of(float) * line number * scannings".
- When the memory area specification that stores data is NULL, the memory area is secured in the function.

Ex.) Navigation Data (scan x 6): real data(6, scan)

status = AMTK_get_SwathFloat(HDF_file_id, data, from_scan, to_scan, access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	Integer	output	1	Normal：The number of acquired scan is returned. Error：A negative value returns when it is not possible to acquire it.
Parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	real	output	1	Memory area where data is stored
from_scan	integer	input	1	Scan number of acquisition start
to_scan	integer	input	1	Scan number of acquisition end
access_lbl	integer	input	1	Please refer to "6.1.1 the HDF access label".

Acquisition of double Type data

The data of double Type is acquired specifying the HDF access label.

- The data of the correspondence is acquired specifying the scanning number.
- The specification of the scanning number specifies the beginning scanning number and the end scanning number.
  The error returns when the end scanning number is smaller than the beginning scanning number.
  When the end scanning number is larger than the scanning number accommodated by HDF Dataset, the data accommodated by HDF Dataset is acquired.
- The user who uses the function has to alocate the memory area where data is stored.
  Size specifies "Number of size of(double) * line number * scannings".
- When the memory area specification that stores data is NULL, the memory area is secured in the function.
  Ex.) Position in Orbit (scan): real*8 data(scan)

status = AMTK_get_SwathDouble(HDF_file_id, data, from_scan, to_scan, access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal:The number of acquired scan is returned. Error:A negative value returns when it is not possible to acquire it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	real8	output	1	Memory area where data is stored
from_scan	integer	input	1	Scan number of acquisition start
to_scan	integer	input	1	Scan number of acquisition end
access_lbl	integer	input	1	Please refer to "6.1.1 the HDF access label".

Acquisition of CharcterType data

The data of CharacterType is acquired specifying the HDF access label.

- The data of the correspondence is acquired specifying the scanning number.
- The specification of the scanning number specifies the beginning scanning number and the end scanning number.
  The error returns when the end scanning number is smaller than the beginning scanning number.
  When the end scanning number is larger than the scanning number accommodated by HDF Dataset, the data accommodated by HDF Dataset is acquired.
- The user who uses the function has to alocate the memory area where data is stored.
  Size specifies "Number of size of(unsigned char) * line number * scannings".
- When the memory area specification that stores data is NULL, the memory area is secured in the function.
  Ex.) SPC Temperature Count (scan x 34): character data(34, scan)

status = AMTK_get_SwathUChar(HDF_file_id, data, from_scan, to_scan, access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal:The number of acquired scan is returned. Error:A negative value returns when it is not possible to acquire it.
Parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	Character	output	1	Memory area where data is stored
from_scan	integer	input	1	Scan number of acquisition start
to_scan	integer	input	1	Scan number of acquisition end
access_lbl	integer	input	1	Please refer to "6.1.1 the HDF access label".

(2) L1A, L1B ,L1R ,L2 ,L3 output function

Product file opening (writing only)				
The HDF product file is opened with write only.				
hid_t AMTK_openH5_Write(char *file_name, int mode)				
Name	Type	IN/OUT	Size	Description
return value				
HDF_file_id	integer	output	1	Normal：HDF access file id Error：A negative value returns when it is not possible to acquire it.
parameter				
file_name	character*500	input	1	AMSR2 HDFfile name
mode	integer	input	1	File open mode AM2_CREATE_MODE : New file The existing file is deleted. AM2_RW_MODE : Overwrite It becomes an error when there is no existing file.

Product file close				
The HDF product file is closed.				
int AMTK_closeH5_Write(hid_t HDF_file_id)				
Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal：0 Error：A negative value returns.
parameter				
HDF_file_id	integer	input	1	HDF access file id

HDF Dataset making				
HDF Dataset of data dimention size is made. — When generating it, HDF Dataset is initialized by the missing value.				
status = AMTK_setDimSize(HDF_file_id, access_lbl, size)				
Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal：0 Error：A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
access_lbl	integer	input	1	Please refer to "6.1.1 the HDF access label".
size(3)	integer	input	1	Dimention size (Three dimention array. For detailed information, Please refer to function Description of AMTK_getDimSize.)

Meta data setting				
The meta data is set by the meta data name of the HDF product.				
status = AMTK_setMetaDataName(HDF_file_id, name, value)				
Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal：0 Error：A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	integer	input	1	HDF access file id
name	character*30	input	1	Meta data name
value	character*10	input	1	Meta data value

Setting at scanning time

Scanning time data (Year, Month, Day, Hour, Minute, second, mili second) is converted to TAI93 real integer and written in the specified scan number.

- The scan number appoints a start scan number and an end scan number.
When an end scan number is smaller than a start scan number, an error returns.
When an end scan number is bigger than a scan number stored away by appointed HDF data set, It acquires data stored away by HDF data set.
- The user using a function allocate memory area storing data.
A data size is "24(=AM2_COMMON_SCANTIME)*dimension size(AMTK_getDimSize function reference)".
Ex.) Scan Time (scan): type(AM2_COMMON_SCANTIME) data(scan)

status = AMTK_setScanTime(HDF_file_id, scan_time, from_scan, to_scan)

Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal:0 Error:A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	integer	input	1	HDF access file id
scan_time	AM2_COMMON_SCANTIME	input	1	Data structure
from_scan	integer	input	1	beginning scanning number
to_scan	integer	input	1	end scanning number

Latitude longitude data setting

The Lat/Long data is written in the specified scan number.

- The scan number appoints a start scan number and an end scan number.
When an end scan number is smaller than a start scan number, an error returns.
When an end scan number is bigger than a scan number stored away by appointed HDF data set, It acquires data stored away by HDF data set.
- The user using a function allocate memory area storing data.
A data size is "8(=AM2_COMMON_LATLON)*dimension size(AMTK_getDimSize function reference)".
Ex.) Latitude of Observation Point for 89A/89B (scan x 486) : type(AM2_COMMON_LATLON) data(486, scan)

status = AMTK_setLatLon(HDF_file_id, latitudelongitude, from_scan, to_scan, access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal:0 Error:A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
latitudelongitude	AM2_COMMON_LATLON	input	1 (x N scan)	Data structure
from_scan	integer	input	1	beginning scanning number
to_scan	integer	input	1	end scanning number
access_lbl	integer	input	1	Please refer to "6.1.1.6 the LATLON function".

Quality information setting

The qualitiy information of L1 data is set.

- The scan number appoints a start scan number and an end scan number.
  When an end scan number is smaller than a start scan number, an error returns.
  When an end scan number is bigger than a scan number stored away by appointed HDF data set, It acquires data stored away by HDF data set.
- The user using a function allocate memory area storing data.
  A data size is “512(AM2_COMMON_LATLON)*dimension size(AMTK_getDimSize function reference)”.
  Ex.) Scan Data Quality (scan x 512) # Data set is char type x sizeof(AMTK_SCAN_DATA_QUALITY)size

: type(AMTK_SCAN_DATA_QUALITY) data(scan)

int AMTK_set_ScanDataQuality(hid_t HDF_file_id, AMTK_SCAN_DATA_QUALITY *quality, int from_scan, int to_scan, int access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	int	output	1	Normal:0 Error:A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
quality	AMTK_SCAN_DATA_QUALITY *	output	1	Data structure
from_scan	int	input	1	beginning scanning number
to_scan	int	input	1	end scanning number
access_lbl	int	input	1	Please refer to "6.1.1 the HDF access label".

Integer Type data setting

Data is set to HDF Dataset as data of integer Type by specifying the HDF access label.

- The data of the correspondence is acquired specifying the scanning number.
- The specification of the scanning number specifies the beginning scanning number and the end scanning number.
  The error returns when the end scanning number is smaller than the beginning scanning number.
  When the end scanning number is larger than the scanning number accommodated by HDF Dataset, the data accommodated by HDF Dataset is acquired.
- The user who uses the function has to alocate the memory area where data is stored.
  Size specifies "Number of size of(int) * line number * scannings".
- When the numerical value that cannot be accommodated in HDF Dataset exists in the setting data, correspondence data is replaced with the error value and it writes it in HDF. And a negative value of the warning is set to return value.

Ex.) Hot Load Count 6 to 36 (12 x scan x 16): integer data(16 * scan * 12)

status = AMTK_set_SwathInt(HDF_file_id, data, from_scan, to_scan, access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal:0 Error:A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	integer	input	1	Memory area where data is stored
from_scan	integer	input	1	beginning scanning number
to_scan	integer	input	1	end scanning number
access_lbl	integer	input	1	Please refer to "6.1.1 the HDF access label".

Floating Type data setting

Data is set to HDF Dataset as data of floating Type by specifying the HDF access label.

- The data of the correspondence is acquired specifying the scanning number.
- The specification of the scanning number specifies the beginning scanning number and the end scanning number.
  The error returns when the end scanning number is smaller than the beginning scanning number.
  When the end scanning number is larger than the scanning number accommodated by HDF Dataset, the data accommodated by HDF Dataset is acquired.
- The user who uses the function has to alocate the memory area where data is stored.
  Size specifies "Number of sizeof(float) * line number * scannings".
- When you convert the value of the floating number into the integral value by the scale factor of HDF Dataset of the correspondence, the first place of the decimal point is rounded off after it divides in the scale factor.

Ex.) Navigation Data (scan x 6): real data(6, scan)

status = AMTK_set_SwathFloat(HDF_file_id, data, from_scan, to_scan, access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal：0 Error：A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	real	input	1	Memory area where data is stored
from_scan	integer	input	1	beginning scanning number
to_scan	integer	input	1	end scanning number
access_lbl	integer	input	1	Please refer to "6.1.1 the HDF access label".

Double Type data setting

Data is set to HDF Dataset as data of double Type by specifying the HDF access label.

- The data of the correspondence is acquired specifying the scanning number.
- The specification of the scanning number specifies the beginning scanning number and the end scanning number.
  The error returns when the end scanning number is smaller than the beginning scanning number.
  When the end scanning number is larger than the scanning number accommodated by HDF Dataset, the data accommodated by HDF Dataset is acquired.
- The user who uses the function has to alocate the memory area where data is stored.
  Size specifies "Number of size of(double) * line number * scannings".

Ex.) Position in Orbit (scan): real*8 data(scan)

status = AMTK_set_SwathDouble(HDF_file_id, data, from_scan, to_scan, access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal：0 Error：A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	real8	input	1	Memory area where data is stored
from_scan	integer	input	1	beginning scanning number
to_scan	integer	input	1	end scanning number
access_lbl	integer	input	1	Please refer to "6.1.1 the HDF access label".

Char Type data setting

Data is set to HDF Dataset as data of Char Type by specifying the HDF access label.

- The data of the correspondence is acquired specifying the scanning number.
- The specification of the scanning number specifies the beginning scanning number and the end scanning number.
  The error returns when the end scanning number is smaller than the beginning scanning number.
  When the end scanning number is larger than the scanning number accommodated by HDF Dataset, the data accommodated by HDF Dataset is acquired.
- The user who uses the function has to alocate the memory area where data is stored.
  Size specifies "Number of sizeof(unsigned char) * line number * scannings".
- When the numerical value that cannot be accommodated in HDF Dataset exists in the setting data, correspondence data is replaced with the error value and it writes it in HDF. And a negative value of the warning is set to return value.
  Ex.) SPC Temperature Count (scan x 34): character data(34, scan)

status = AMTK_set_SwathUChar(HDF_file_id, data, from_scan, to_scan, access_lbl)

Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal：0 Error：A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	character	input	1	Memory area where data is stored
from_scan	integer	input	1	beginning scanning number
to_scan	integer	input	1	end scanning number
access_lbl	integer	input	1	Please refer to "6.1.1 the HDF access label".

5.2.2 Input functions

(1) L1A, L1B ,L1R function

There is no input function peculiar to L1 because the common function is used.

(2) L2 input function

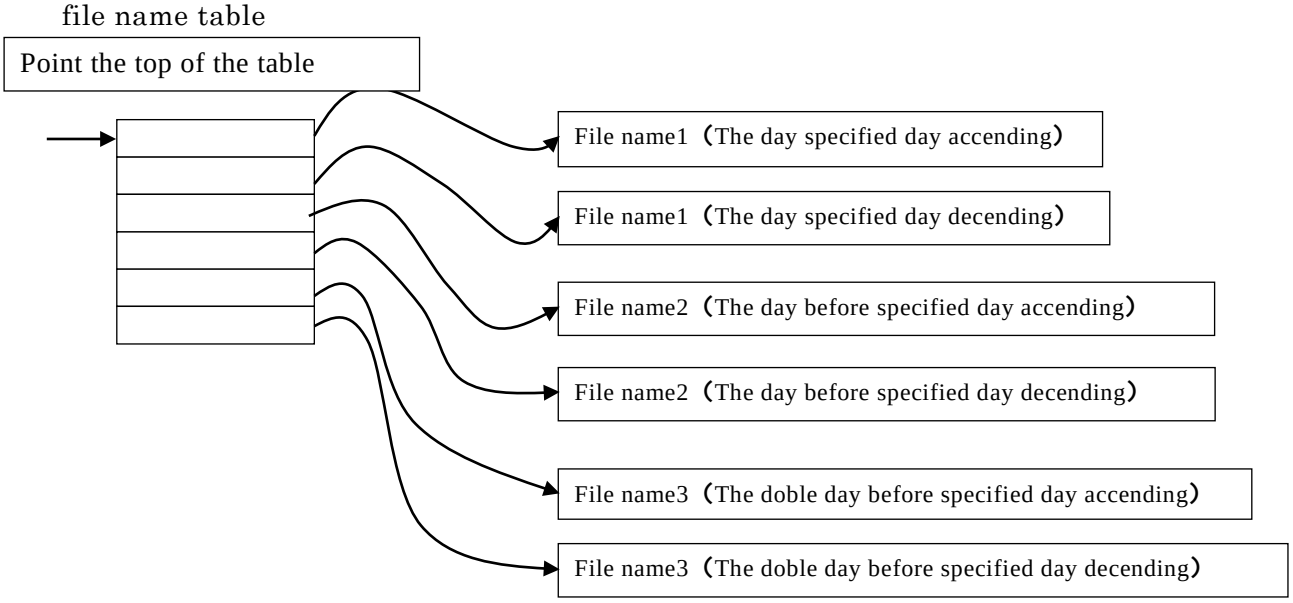
There is no input function peculiar to L2 because the common function is used.

(3) L3 input function

Acquistion of Integer Type data				
The data of integer Type is acquired specifying the HDF access label. - All of the array data of the L3 product specified with the access label are acquired. To get the information of array Size, it uses the AMTK_getDimSize function. - The user who uses the function secures Memory area where data is stored. Size specifies "4*Dimension". Ex.) Average Number (V) (Xpixel x Ypixel): integer data(Ypixel * Xpixel)				
status = AMTK_get_GridInt(HDF_file_id, data, access_lbl)				
Name	Type	IN/OUT	Size	Description
Return value				
status	integer	output	1	Normal：The number of acquired scan is returned. Error：A negative value returns when it is not possible to acquire it.
Parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	integer	output	1	Memory area where data is stored
access_lbl	integer	input	1	Please refer to "6.1.1 the HDF access label".

Acquistion of float type data **				
The data of float Type is acquired specifying the HDF access label. - All of the array data of the L3 product specified with the access label are acquired. To get the information of array Size, it uses the AMTK_getDimSize function. - The user who uses the function secures Memory area where data is stored. Size specifies "4*Dimension". Ex.) Standard Deviation (V) (Xpixel x Ypixel): real data(Ypixel * Xpixel)				
status = AMTK_get_GridFloat(HDF_file_id, data, access_lbl)				
Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal：The number of acquired scan is returned. Error：A negative value returns when it is not possible to acquire it.
Parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	real	output	1	Memory area where data is stored
access_lbl	integer	input	1	Please refer to "6.1.1 the HDF access label".

Average data acquisition on the third				
The average data of three days of level 3 is acquired. In the case the input file is not based on rules, the error number (-250 or more) is returned. Please refer to ”7 Error Number ID” for the details.				
status = AMTK_get_Grid03D(average, name_table)				
Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal：0 Error：A negative value returns when it is not possible to acquire it.
parameter				
average	real	output	M pixcel x N scan	Data structure 0.1-degree or 0.25-degree lattice are judged with file name. The specified file specifies all files of the same lattice.
name_table	character*30	input	256	File name table (Please refer to the figure below).



5.2.3 Output function

(1) L1 output functions

There is no input function peculiar to L1 because the common function is used.

(2) L2 output function

There is no input function peculiar to L2 because the common function is used.

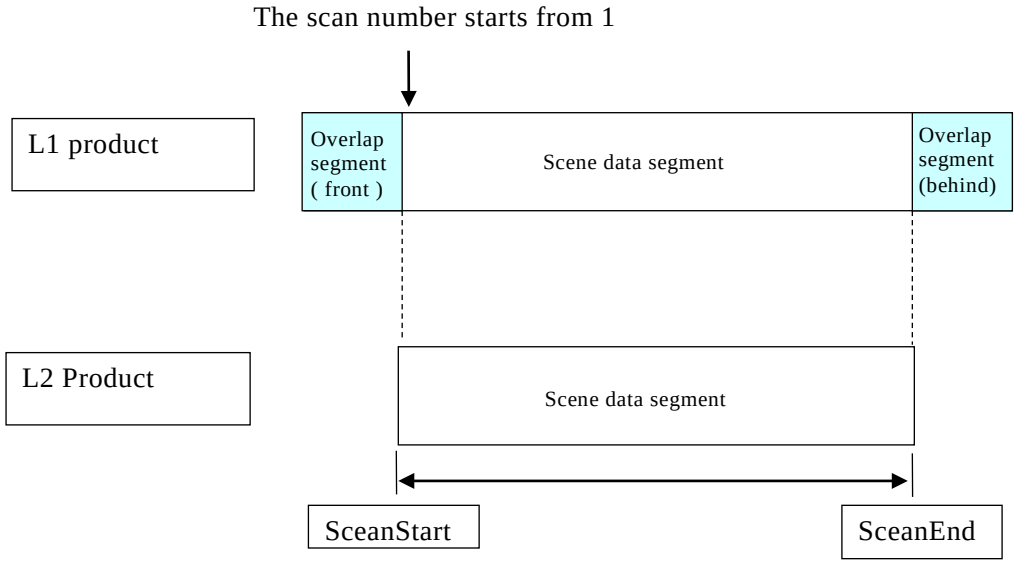
(3) L3 output function

Acquistion of Integer Type data				
The data of integer Type is acquired specifying the HDF access label. - All of the array data of the L3 product specified with the access label are acquired. To get the information of array Size, it uses the AMTK_getDimSize function. - The user who uses the function secures Memory area where data is stored. Size specifies "4*Dimension". Ex.) Average Number (V) (Xpixel x Ypixel): integer data(Ypixel * Xpixel)				
status = AMTK_set_GridInt(HDF_file_id, data, access_lbl)				
Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal：0 Error：A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	integer	input	1	Memory area where data is stored
access_lbl	integer	input	1	Please refer to "6.1.1 the HDF access label".

Acquistion of float Type data				
The data of float Type is acquired specifying the HDF access label. - All of the array data of the L3 product specified with the access label are acquired. To get the information of array Size, it uses the AMTK_getDimSize function. - The user who uses the function secures Memory area where data is stored. Size specifies "4*Dimension". Ex.) Standard Deviation (V) (Xpixel x Ypixel): real data(Ypixel * Xpixel)				
status = AMTK_set_GridFloat(HDF_file_id, data, access_lbl)				
Name	Type	IN/OUT	Size	Description
return value				
status	integer	output	1	Normal：0 Error：A negative value returns when it is not possible to set it.
parameter				
HDF_file_id	hid_t	input	1	HDF access file id
data	real	input	1	Memory area where data is stored
access_lbl	integer	input	1	Please refer to "6.1.1 the HDF access label".

5.3 Scan number

The AMTK function executes to in/out-put with pointing the scan number. The scan number starts from 1 on L1 and L2 product. About overlap segment to be included in a L1 product, it is necessary to make a scan number of the department of scene data agree in L1 and L2. Show the details in the following figure.



The normal access is taken for scene data segment only. In the case of a user accesses an overlap segment with a L1 product, use the following values.

Overlap segment (front) : negative number ~ 0
Overlap segment (behind) : SeanEnd＋Overlap

The SceanStart and SceanEnd of the Scene data segment is defined as OverlapScans and NumberOfScans.

SceanStart : OverlapScans＋1
SceamEnd : OverlapScans＋NumberOfScans

For all scan segment, the number of the scan is defined as follows.

Scean scan number : NumberOfScans
Product scan number : NumberOfScans＋OverlapScans× 2

As the L2 product, the value of the Scean scan number is the same as the Product scan number.As the L1 product, the value of the Scean scan number is not the same as the Product scannumber so that there is the Overlap segments.

If the scan number set to AMTK functions is outside of the coverage of the Product scan number, it will be error.

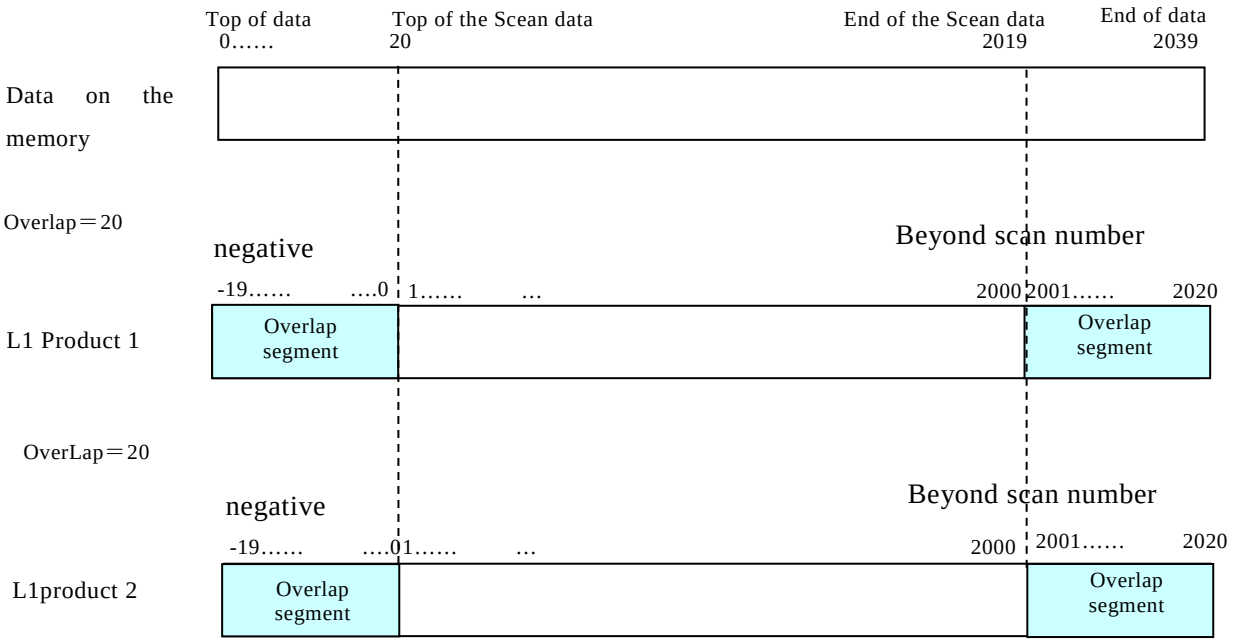
The case that L1 input to L1 output (their overlap scan are same) and the case t

hat L1 input to L2 output (their overlap scan are not same) are shown as followe d next page.

Input L1 ・Output L1

read : -19～2020

write : -19～2020　There is the data position on the memory in the top of data.

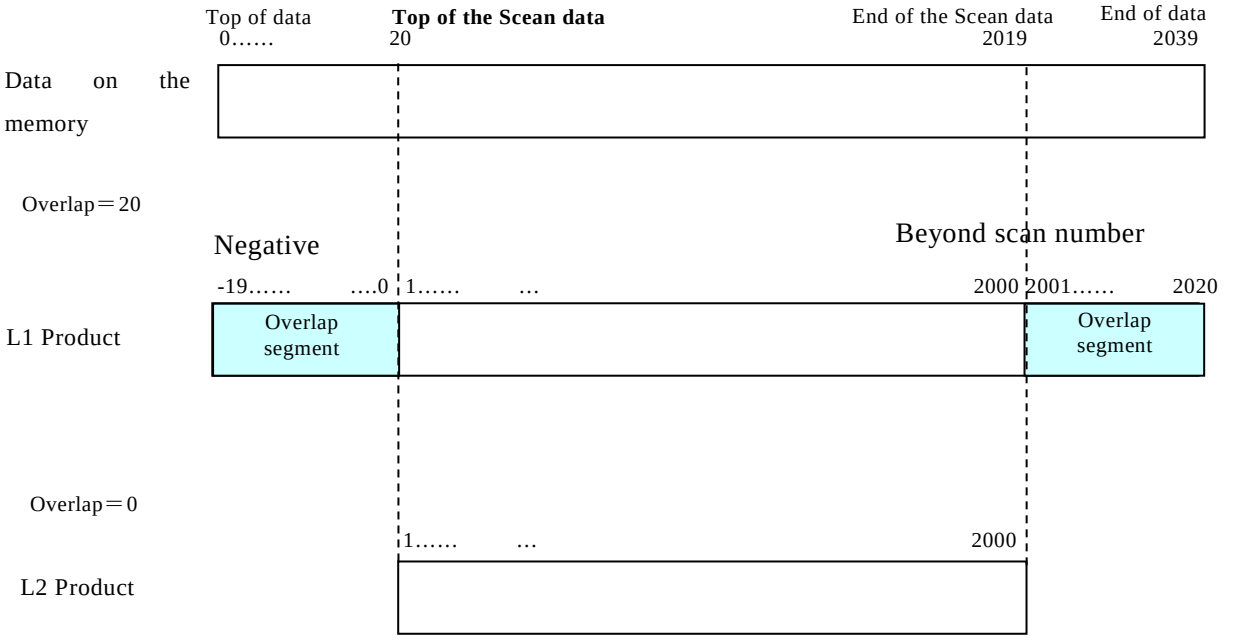


Input L1・Output L2

Read : -19～2020

Write : 1～2000

The data position of the memory to write in appoints the scene data top.



* The overlap scan number refers to the value of OverlapScans of Metadata. The re aboveexample assuming the following information.
  - L1 Product : OverlapScans = 20
  - L2 Product : OverlapScans = 0

5.4 Stored value

The AMTK functions are classified by data type. On the other hand, the data stored in HDF are classified, too. Note the case that the type of AMTK function is not the type of the data stored in HDF.

Show the following the value that is stored in HDF when data are made first and the valu stored when data in/out-put.

The Dataset are made with AMTK_setDimSize function. This function makes the Dataset with the appointed size and store the following value into HDF as initial value.

Type of C-Language	Type of Fortran	Stored value	Remarks
Int	Integer*4	-32766	
unsigned int	-	65533	
float	real*4	-9997.0	
double	real*8	-9997.0	
unsihned char	-	253	

The value stored in the case of input and output is fixed by a function and a combination with the data type of the product to use. There are deficit value and an abnormal value for data to use. When there are not data in the appointed place, the deficit value sets it. When there are the abnormality value, the abnormal value sets parity error. Deficit value, neither of the abnormal value process the scale factor to become a fixed price. Show below a function and the combination of the data type of the product and deficit value, an abnormal value. The upper section of each column, deficit value, the lower berth show an abnormal value. Show the data type with the model of the C language.

Type of C-Language	Int()	Float()	Double()	UChar	Remarks
Int	-32768	-32768.0	—	—	
	-32767	-32767.0			
unsigned int	65535	65535.0	—	—	
	65534	65534.0			
float	—	-9999.0	—	—	
		-9998.0			
double	—	—	-9999.0	—	
			-9998.0		
unsihned char	255	—	255.0	255	
	254		254.0		

6 In/Out-put data

6.1 Data definition

6.1.1 HDF access lavel

The function of AMTK is made universally, and a number is given to each data set to specify the data set that is an access unit of HDF.

A user appoints a direct number or uses the identifier defined as an access label shown in the following table. When it in/out put the data set with appointing the access lave, it should refer the following medata at the function of AMTK. Therefore, it is neccssary to store an appropriate value for meta data beforehand. When there is no meta data, it returns an error number of the peculiarity every each meta data.

Refer to "7 Error Number ID" for the details.

Meta data	description
ProductName	It is referred to distinguish product classification.
OverlapScans	It is referred to specify the scan number of the input and output object. (Refer to "5.3 Scan number"abut the scan number.) It is not referred in the case of product classification is L3.
GeophysicalName	In the case of product classification is L2 or L3, it is reffered to distinguish Geophysical quantity. (Refer to "2.5 Geophysical quantity definition file"about the geophysical quantity.)
GranuleID	In the case of L3, it is referred to distinguish product classification.
CoRegistrationParameterA1	In the case of appointing the following access label, it is referred to calculate a low frequency Lat/Long. AM2_LATLON_06, AM2_LATLON_07, AM2_LATLON_10,AM2_LATLON_18, AM2_LATLON_23, AM2_LATLON_36, AM2_LATLON_LO
CoRegistrationParameterA2	
Projection	In the case of appointing "AM2_GRID_LATLON" as access label, it is referred to distinguish a map projection system.
OrbitDirection	When it runs AMTK_get_Grid03D(), it is referred to distinguishaorbital direction.
ProductionDateTime	When it runs AMTK_get_Grid03D(), it is referred to get a production data time.

The following table shows the details of the contents in the following page.

Data name	The name of HDF file data set.
Type	It is data type stored by HDF and the argument data type of the access function.
Access function	There are access functions for input and output with. Select thefu nction desired. "-" indicates that the target function is not availa ble. Be careful for there are different cases in a data type stored by HDF anda access function. Even if integer value data is stor ed away by data set, there is a case to become the access functi on of the true numberical value to consider Scale Facter.
Access label	It is an identifier to access HDF data set. (Alias definition of the access ID)
Access ID	A number for access to HDF data set.
Remarks	Refernece information. (Ex. Valu of Scale Facter)

6.1.1.1 L1A

	Data name	Type(HDF)	Type	Access function		Access label	Access ID	Remark
				In	Out			
1	Product Meta Data							
2	Scan Time	double	AM2_COMMON_SCANTIME	AMTK_getScanTime	AMTK_setScanTime	AM2_SCAN_TIME_DEF	10010	
3	Position in Orbit	double	double	AMTK_get_SwathDouble	AMTK_set_SwathDouble	AM2_POS_ORBIT	10020	
4	Navigation Data	float	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_NAVI	10030	
5	Attitude Data	float	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_ATT	10040	
6	Observation Count (6.9GHz,V)	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OC06V	10050	
7	Observation Count (6.9GHz,H)	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OC06H	10060	
8	Observation Count (7.3GHz,V)	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OC07V	10070	
9	Observation Count (7.3GHz,H)	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OC07H	10080	
10	Observation Count (10.7GHz,V)	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OC10V	10090	
11	Observation Count (10.7GHz,H)	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OC10H	10100	
12	Observation Count (18.7GHz,V)	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OC18V	10110	
13	Observation Count (18.7GHz,H)	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OC18H	10120	
14	Observation Count (23.8GHz,V)	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OC23V	10130	
15	Observation Count (23.8GHz,H)	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OC23H	10140	
16	Observation Count (36.5GHz,V)	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OC36V	10150	
17	Observation Count (36.5GHz,H)	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OC36H	10160	
18	Observation Count (89.0GHz-A,V)	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OC89AV	10170	
19	Observation Count (89.0GHz-A,H)	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OC89AH	10180	
20	Observation Count (89.0GHz-B,V)	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OC89BV	10190	
21	Observation Count (89.0GHz-B,H)	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OC89BH	10200	
22	Observation Count (6.9GHz-36.5GHz, V&H)	signed int	signed int	AMTK_get_SwathInt	—	AM2_OC_LO	10210	
23	Observation Count (89GHz-A, V&H)	signed int	signed int	AMTK_get_SwathInt	—	AM2_OC_89A	10220	
24	Observation Count (89GHz-B, V&H)	signed int	signed int	AMTK_get_SwathInt	—	AM2_OC_89B	10230	

	Data name	Type(HDF)	Type	Access function		Access label	Access ID	Remark
				In	Out			
25	Hot Load Count 6 to 36	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_HTS_LO	10240	
26	Hot Load Count 89	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_HTS_HI	10250	
27	Cold Sky Mirror Count 6 to 36	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_CSM_LO	10260	
28	Cold Sky Mirror Count 89	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_CSM_HI	10270	
29	Rx Offset_Gain Count	unsigned int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OFF_GAIN	10290	
30	Latitude of Observation Point for 89A	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
31	Longitude of Observation Point for 89A	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
32	Latitude of Observation Point for 89B	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
33	Longitude of Observation Point for 89B	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
34	Sun Azimuth	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_SUN_AZ	10300	
35	Sun Elevation	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_SUN_EL	10310	
36	Earth Incidence	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_EARTH_INC	10320	
37	Earth Azimuth	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_EARTH_AZ	10330	
38	Land_Ocean Flag 6 to 36	unsigned char	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_LOF_LO	10340	
39	Land_Ocean Flag 89	unsigned char	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_LOF_HI	10350	
40	Observation Supplement	binary (2byte)	unsigned char *2	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_OB_SPL	10420	
41	SPC Temperature Count	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_SPC_TEMP	10430	
42	SPS Temperature Count	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_SPS_TEMP	10440	
43	PCD Data	binary (64byte)	unsigned char *64	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PCD	10450	
44	Scan Data Quality	binary (512byte)	512byte を float *128 に分割	AMTK_getScanDataQuality	AMTK_setScanDataQuality	AM2_SCAN_QUAL	10460	

	Data name	Type(HDF)	Type	Access function		Access label	Access ID	Remark
				In	Out			
45	Pixel Data Quality 6 to 36	binary (2byte)	1bit を unsigned char に変換 (unsigned char*16 になる)	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_LO	10470	
46	Pixel Data Quality 89	binary (1byte)	1bit を unsigned char に変換 (unsigned char*8 になる)	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_HI	10480	
47	Interpolation Flag 6 to 36	binary (1byte)	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_INTPL_LO	10490	
48	Interpolation Flag 89	binary (1byte)	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_INTPL_HI	10500	

6.1.1.2 L1B

	Data name	Type(HDF)	Type	Access function		Access label	Access ID	Remark
				In	Out			
1	Product Meta Data	-					-	
2	Scan Time	double	AM2_COMMON_SCANTIME	AMTK_getScanTime	AMTK_setScanTime	AM2_SCAN_TIME_DEF	10010	
3	Position in Orbit	double	double	AMTK_get_SwathDouble	AMTK_set_SwathDouble	AM2_POS_ORBIT	10020	
4	Navigation Data	float	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_NAVI	10030	
5	Attitude Data	float	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_ATT	10040	
6	Brightness Temperature (6.9GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB06V	11050	
7	Brightness Temperature (6.9GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB06H	11060	
8	Brightness Temperature (7.3GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB07V	11070	
9	Brightness Temperature (7.3GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB07H	11080	
10	Brightness Temperature (10.7GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB10V	11090	
11	Brightness Temperature (10.7GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB10H	11100	
12	Brightness Temperature (18.7GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB18V	11110	
13	Brightness Temperature (18.7GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB18H	11120	
14	Brightness Temperature (23.8GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB23V	11130	
15	Brightness Temperature (23.8GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB23H	11140	
16	Brightness Temperature (36.5GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB36V	11150	
17	Brightness Temperature (36.5GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB36H	11160	
18	Brightness Temperature (89.0GHz-A,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB89AV	11170	
19	Brightness Temperature (89.0GHz-A,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB89AH	11180	
20	Brightness Temperature (89.0GHz-B,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB89BV	11190	
21	Brightness Temperature (89.0GHz-B,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB89BH	11200	
22	Brightness Temperature (6.9GHz-36.5GHz, V&H)	unsigned int	float	AMTK_get_SwathFloat	—	AM2_TB_LO	11210	
23	Brightness Temperature (89GHz-A, V&H)	unsigned int	float	AMTK_get_SwathFloat	—	AM2_TB_89A	11220	
24	Brightness Temperature (89GHz-B, V&H)	unsigned int	float	AMTK_get_SwathFloat	—	AM2_TB_89B	11230	

	Data name	Type(HDF)	Type	Access function		Access label	Access ID	Remark
				In	Out			
25	Hot Load Count 6 to 36	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_HTS_LO	10240	
26	Hot Load Count 89	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_HTS_HI	10250	
27	Cold Sky Mirror Count 6 to 36	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_CSM_LO	10260	
28	Cold Sky Mirror Count 89	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_CSM_HI	10270	
29	Rx Offset_Gain Count	unsigned int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_OFF_GAIN	10290	
30	Latitude of Observation Point for 89A	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
31	Longitude of Observation Point for 89A	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
32	Latitude of Observation Point for 89B	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
33	Longitude of Observation Point for 89B	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
34	Sun Azimuth	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_SUN_AZ	10300	
35	Sun Elevation	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_SUN_EL	10310	
36	Earth Incidence	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_EARTH_INC	10320	
37	Earth Azimuth	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_EARTH_AZ	10330	
38	Land_Ocean Flag 6 to 36	unsigned char	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_LOF_LO	10340	
39	Land_Ocean Flag 89	unsigned char	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_LOF_HI	10350	
40	Observation Supplement	binary (2byte)	unsigned char *2	AMTK_set_SwathUChar	AMTK_set_SwathUChar	AM2_OB_SPL	10420	
41	SPC Temperature Count	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_SPC_TEMP	10430	
42	SPS Temperature Count	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_SPS_TEMP	10440	
43	PCD Data	binary (64byte)	unsigned char *64	AMTK_set_SwathUChar	AMTK_set_SwathUChar	AM2_PCD	10450	
44	Scan Data Quality	binary (512byte)	Convert from 512byte to float *128	AMTK_getScanDataQuality	AMTK_setScanDataQuality	AM2_SCAN_QUAL	10460	
45	Pixel Data Quality 6 to 36	binary (2byte)	Convert from 1bit to unsigned char(unsigned char*16)	AMTK_set_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_LO	10470	

	Data name	Type(HDF)	Type	Access function		Access label	Access ID	Remark
				In	Out			
46	Pixel Data Quality 89	binary (1byte)	Convert from 1bit to unsigned char(unsigned char*16)	AMTK_set_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_HI	10480	
47	Interpolation Flag 6 to 36	binary (1byte)	unsigned char	AMTK_set_SwathUChar	AMTK_set_SwathUChar	AM2_INTPL_LO	10490	
48	Interpolation Flag 89	binary (1byte)	unsigned char	AMTK_set_SwathUChar	AMTK_set_SwathUChar	AM2_INTPL_HI	10500	

6.1.1.3 L1R

	Data name	Type(HDF)	Type	Access function		Access label	Access ID	Remark
				In	Out			
1	Product Meta Data	-					-	
2	Scan Time	double	AM2_COMMON_SCANTIME	AMTK_getScanTime	AMTK_setScanTime	AM2_SCAN_TIME_DEF	10010	
3	Position in Orbit	double	double	AMTK_get_SwathDouble	AMTK_set_SwathDouble	AM2_POS_ORBIT	10020	
4	Navigation Data	float	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_NAVI	10030	
5	Attitude Data	float	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_ATT	10040	
	<6GHz resolution>							
6	Brightness Temperature (res06,6.9GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES06_TB06V	12050	
7	Brightness Temperature (res06,6.9GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES06_TB06H	12060	
8	Brightness Temperature (res06,7.3GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES06_TB07V	12070	
9	Brightness Temperature (res06,7.3GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES06_TB07H	12080	
10	Brightness Temperature (res06,10.7GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES06_TB10V	12090	
11	Brightness Temperature (res06,10.7GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES06_TB10H	12100	
12	Brightness Temperature (res06,18.7GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES06_TB18V	12110	
13	Brightness Temperature (res06,18.7GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES06_TB18H	12120	
14	Brightness Temperature (res06,23.8GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES06_TB23V	12130	
15	Brightness Temperature (res06,23.8GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES06_TB23H	12140	
16	Brightness Temperature (res06,36.5GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES06_TB36V	12150	
17	Brightness Temperature (res06,36.5GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES06_TB36H	12160	
18	Brightness Temperature (res06,89.0GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES06_TB89V	12170	
19	Brightness Temperature	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES06_TB89	12180	

	Data name	Type(HDF)	Type	Access function		Access label	Access ID	Remark
				In	Out			
	(res06,89.0GHz,H)					H		
20	Brightness Temperature (res06,all)	unsigned int	float	AMTK_get_SwathFloat	—	AM2_RES06_TB_ALL	12190	
	<10GHz resolution>							
21	Brightness Temperature (res10,10.7GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES10_TB10V	12200	
22	Brightness Temperature (res10,10.7GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES10_TB10H	12210	
23	Brightness Temperature (res10,18.7GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES10_TB18V	12220	
24	Brightness Temperature (res10,18.7GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES10_TB18H	12230	
25	Brightness Temperature (res10,23.8GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES10_TB23V	12240	
26	Brightness Temperature (res10,23.8GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES10_TB23H	12250	
27	Brightness Temperature (res10,36.5GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES10_TB36V	12260	
28	Brightness Temperature (res10,36.5GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES10_TB36H	12270	
29	Brightness Temperature (res10,89.0GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES10_TB89V	12280	
30	Brightness Temperature (res10,89.0GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES10_TB89H	12290	
31	Brightness Temperature (res10,all)	unsigned int	float	AMTK_get_SwathFloat	—	AM2_RES10_TB_ALL	12300	
	<23GHz resolution>							
32	Brightness Temperature (res23,18.7GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES23_TB18V	12310	
33	Brightness Temperature (res23,18.7GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES23_TB18H	12320	
34	Brightness Temperature (res23,23.8GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES23_TB23V	12330	
35	Brightness Temperature (res23,23.8GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES23_TB23H	12340	
36	Brightness Temperature (res23,36.5GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES23_TB36V	12350	
37	Brightness Temperature (res23,36.5GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES23_TB36H	12360	
38	Brightness Temperature (res23,89.0GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES23_TB89V	12370	
39	Brightness Temperature	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES23_TB89	12380	

	Data name	Type(HDF)	Type	Access function		Access label	Access ID	Remark
				In	Out			
	(res23,89.0GHz,H)					H		
40	Brightness Temperature (res23,all)	unsigned int	float	AMTK_get_SwathFloat	—	AM2_RES23_TB_ALL	12390	
	<36GHz resolution>							
41	Brightness Temperature (res36,36.5GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES36_TB36V	12400	
42	Brightness Temperature (res36,36.5GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES36_TB36H	12410	
43	Brightness Temperature (res36,89.0GHz,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES36_TB89V	12420	
44	Brightness Temperature (res36,89.0GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES36_TB89H	12430	
45	Brightness Temperature (res36,all)	unsigned int	float	AMTK_get_SwathFloat	—	AM2_RES36_TB_ALL	12440	
	<89GHz resolution>							
46	Brightness Temperature (original,89GHz-A,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB89AV	11170	
47	Brightness Temperature (original,89GHz-A,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB89AH	11180	
48	Brightness Temperature (original,89GHz-B,V)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB89BV	11190	
49	Brightness Temperature (original,89GHz-B,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_TB89BH	11200	
50	Brightness Temperature (original,89GHz-A,V&H)	unsigned int	float	AMTK_get_SwathFloat	—	AM2_TB_89A	11220	
51	Brightness Temperature (original,89GHz-B,V&H)	unsigned int	float	AMTK_get_SwathFloat	—	AM2_TB_89B	11230	
52	Latitude of Observation Point for 89A	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
53	Longitude of Observation Point for 89A	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
54	Latitude of Observation Point for 89B	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
55	Longitude of Observation Point for 89B	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
56	Area Mean Height	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_MEAN_HEIGHT	12510	
57	Sun Azimuth	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_SUN_AZ	10300	
58	Sun Elevation	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_SUN_EL	10310	

	Data name	Type(HDF)	Type	Access function		Access label	Access ID	Remark
				In	Out			
59	Earth Incidence	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_EARTH_INC	10320	
60	Earth Azimuth	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_EARTH_AZ	10330	
61	Land_Ocean Flag 6 to 36	unsigned char	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_LOF_RES_LO	12560	
62	Land_Ocean Flag 89	unsigned char	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_LOF_RES_HI	12570	
63	Scan Data Quality	binary (512byte)	Convert form 512byte to float *128	AMTK_getScanDataQuality	AMTK_setScanDataQuality	AM2_SCAN_QUAL	10460	Int or Float
64	Pixel Data Quality 6 to 36	binary (2byte)	Convert form 1bitto unsigned char(unsigned char*16)	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_LO	10470	Refer to 6.1.1.7
65	Pixel Data Quality 89	binary (1byte)	Convert form 1bitto unsigned char(unsigned char*16)	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_HI	10480	
				AMTK_get_SwathInt	AMTK_set_SwathInt			

6.1.1.4 L2

	Data name	Type(HDF)	Type	Access function		Access label	Access ID	Remark
				In	Out			
Low resolution								
1	Product Meta Data	-					-	
2	Scan Time	double	AM2_COMMON_SCANTIME	AMTK_getScanTime	AMTK_setScanTime	AM2_SCAN_TIME_DEF	10010	
3	Position in Orbit	double	double	AMTK_get_SwathDouble	AMTK_set_SwathDouble	AM2_POS_ORBIT	10020	
4	Geophysical Data	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_SWATH_GEO1	21030	Refert to Note 1 and 6.1.1.7
5						AM2_SWATH_GEO2	21040	
6						AM2_SWATH_GEO3	21050	
7						AM2_SWATH_GEOA	21060	
8	Latitude of Observation Point	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
9	Longitude of Observation Point	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
10	Pixel Data Quality	binary (1byte)	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL	21070	Refert to 6.1.1.7
High resolution								
1	Product Meta Data	-				-	-	
2	Scan Time	double	double	AMTK_get_SwathDouble	AMTK_set_SwathDouble	AM2_SCAN_TIME_DEF	10010	
3	Position in Orbit	double	double	AMTK_get_SwathDouble	AMTK_set_SwathDouble	AM2_POS_ORBIT	10020	
4	Geophysical Data for 89A	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_SWATHA_GEO1	22030	Refert to Note 1 and 6.1.1.7
5						AM2_SWATHA_GEO2	22040	
6						AM2_SWATHA_GEO3	22050	
7						AM2_SWATHA_GEOA	22060	
8	Geophysical Data for 89B	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_SWATHB_GEO1	22070	Refert to

	Data name	Type(HDF)	Type	Access function		Access label	Access ID	Remark
				In	Out			
9						AM2_SWATHB_GEO2	22080	Note 1 and 6.1.1.7
10						AM2_SWATHB_GEO3	22090	
11						AM2_SWATHB_GEOA	22100	
12	Latitude of Observation Point for 89A	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
13	Longitude of Observation Point for 89A	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
14	Latitude of Observation Point for 89B	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
15	Longitude of Observation Point for 89B	float	float	Refer to LATLON function	Refer to LATLON function	Refer to LATLON function	-	
16	Pixel Data Quality for 89A	binary (1byte)	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_A	22110	Refert to 6.1.1.7
17	Pixel Data Quality for 89B	binary (1byte)	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_B	22120	Refert to 6.1.1.7

Note 1 : show the Scale factors

Physical amount			scale factor
Name	Mark	unit	
Integrated Water vapor	TPW	kg/m2	0.01
Integrated Cloud liquid Water	CLW	kg/m2	0.001
Sea Surface Winds	SSW	m/s	0.01
Sea Surface Temperature	SST	℃	0.01
Sea Ice Concentration	SIC	%	0.1
Snow Depth	SND	cm	0.1
Soil Moisture	SMC	%	0.1
Precipitation Rate	PRC	mm/h	0.01

6.1.1.5 L3

Daily (High resolution)

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
EQR, brightness temperature								
1	ProductMeta Data	-					-	
2	Brightness Temperature (V)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV	31010	
3	Brightness Temperature (H)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH	31020	
4	Brightness Temperature (V&H)	signed int	float	AMTK_get_GridFloat	—	AM2_GRID_TB	31030	
5	Time Information	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TIME	31040	
PS The North, Brightness temperature								
1	ProductMeta Data	-					-	
2	Brightness Temperature (V)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV	31010	
3	Brightness Temperature (H)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH	31020	
4	Brightness Temperature (V&H)	signed int	float	AMTK_get_GridFloat	—	AM2_GRID_TB	31030	
5	Time Information	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TIME	31040	
PS The south, Brightness temperature								
1	ProductMeta Data	-					-	
2	Brightness Temperature (V)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV	31010	
3	Brightness Temperature (H)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH	31020	
4	Brightness Temperature (V&H)	signed int	float	AMTK_get_GridFloat	—	AM2_GRID_TB	31030	
5	Time Information	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TIME	31040	
EQR, physical quantity								
1	ProductMeta Data	-					-	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	Refert to Note 1 and 6.1.1.7
3						AM2_GRID_GEO2	31320	
4						AM2_GRID_GEO3	31330	

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
5						AM2_GRID_GEOA	31340	
6	Time Information	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TIME	31040	
PS The north, Sea Ice Concentration								
1	ProductMeta Data	-					-	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	Refert to Note 1 and 6.1.1.7
3						AM2_GRID_GEO2	31320	
4						AM2_GRID_GEO3	31330	
5						AM2_GRID_GEOA	31340	
6	Time Information	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TIME	31040	
PS The south, Sea Ice Concentration								
1	ProductMeta Data	-					-	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	Refert to Note 1 and 6.1.1.7
3						AM2_GRID_GEO2	31320	
4						AM2_GRID_GEO3	31330	
5						AM2_GRID_GEOA	31340	
6	Time Information	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TIME	31040	
PS The north, Snow Depth								
1	ProductMeta Data	-					-	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	Refert to 6.1.1.7
3						AM2_GRID_GEO2	31320	
4						AM2_GRID_GEO3	31330	
5						AM2_GRID_GEOA	31340	
6	Time Information	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TIME	31040	

Daily (Low resolution)

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
EQR, Brightness temperature								
1	ProductMeta Data	-					-	
2	Brightness Temperature (V)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV	31010	
3	Brightness Temperature (H)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH	31020	
	Brightness Temperature (V&H)	unsigned int	float	AMTK_get_GridFloat	—	AM2_GRID_TB	31030	
4	Time Information	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TIME	31040	
PS The north, Brightness temperature								
1	ProductMeta Data	-					-	
2	Brightness Temperature (V)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV	31010	
3	Brightness Temperature (H)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH	31020	
	Brightness Temperature (V&H)	unsigned int	float	AMTK_get_GridFloat	—	AM2_GRID_TB	31030	
4	Time Information	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TIME	31040	
PS Southern, Brightness temperature								
1	ProductMeta Data	-					-	
2	Brightness Temperature (V)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV	31010	
3	Brightness Temperature (H)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH	31020	
	Brightness Temperature (V&H)	unsigned int	float	AMTK_get_GridFloat	—	AM2_GRID_TB	31030	
4	Time Information	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TIME	31040	
EQR, physical quantity								
1	ProductMeta Data	-					-	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	Refert to Note 1 and 6.1.1.7
3						AM2_GRID_GEO2	31320	
4						AM2_GRID_GEO3	31330	

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
5						AM2_GRID_GEOA	31340	
6	Time Information	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TIME	31040	
PS The north, Sea Ice Concentration								
1	ProductMeta Data	-					-	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	Refert to Note 1 and 6.1.1.7
3						AM2_GRID_GEO2	31320	
4						AM2_GRID_GEO3	31330	
5						AM2_GRID_GEOA	31340	
6	Time Information	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TIME	31040	
PS The south, Sea Ice Concentration								
1	ProductMeta Data	-					-	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	Refert to Note 1 and 6.1.1.7
3						AM2_GRID_GEO2	31320	
4						AM2_GRID_GEO3	31330	
5						AM2_GRID_GEOA	31340	
6	Time Information	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TIME	31040	
PS The north, Snow Depth								
1	ProductMeta Data	-					-	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	Refert to Note 1 and 6.1.1.7
3						AM2_GRID_GEO2	31320	
4						AM2_GRID_GEO3	31330	
5						AM2_GRID_GEOA	31340	

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
6	Time Information	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TIME	31040	

Monthly (High resolution)

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
EQR, Brightness temperature								
1	ProductMeta Data	-					-	
2	Brightness Temperature (V)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV	31010	
3	Brightness Temperature (H)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH	31020	
4	Brightness Temperature (V&H)	unsigned int	float	AMTK_get_GridFloat	—	AM2_GRID_TB	31030	
5	Standard Deviation (V)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV_STD	34140	
6	Standard Deviation (H)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH_STD	34150	
7	Standard Deviation (V&H)	signed int	float	AMTK_get_GridFloat	—	AM2_GRID_TB_STD	34160	
8	Average Number (V)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBV_ANUM	34170	
9	Average Number (H)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBH_ANUM	34180	
10	Average Number (V&H)	signed int	signed int	AMTK_get_GridInt	—	AM2_GRID_TB_ANUM	34190	
11	Total Number (V)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBV_TNUM	34200	
12	Total Number (H)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBH_TNUM	34210	
13	Total Number (V&H)	signed int	signed int	AMTK_get_GridInt	—	AM2_GRID_TB_TNUM	34220	

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
PS The north, Brightness temperature								
1	ProductMeta Data	-					-	
2	Brightness Temperature (V)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV	31010	
3	Brightness Temperature (H)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH	31020	
4	Brightness Temperature (V&H)	unsigned int	float	AMTK_get_GridFloat	—	AM2_GRID_TB	31030	
5	Standard Deviation (V)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV_STD	34140	
6	Standard Deviation (H)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH_STD	34150	
7	Standard Deviation (V&H)	signed int	float	AMTK_get_GridFloat	—	AM2_GRID_TB_STD	34160	
8	Average Number (V)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBV_ANUM	34170	
9	Average Number (H)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBH_ANUM	34180	
10	Average Number (V&H)	signed int	signed int	AMTK_get_GridInt	—	AM2_GRID_TB_ANUM	34190	
11	Total Number (V)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBV_TNUM	34200	
12	Total Number (H)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBH_TNUM	34210	
13	Total Number (V&H)	signed int	signed int	AMTK_get_GridInt	—	AM2_GRID_TB_TNUM	34220	

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
PS The south, Brightness temperature								
1	ProductMeta Data	-					-	
2	Brightness Temperature (V)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV	31010	
3	Brightness Temperature (H)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH	31020	
4	Brightness Temperature (V&H)	unsigned int	float	AMTK_get_GridFloat	—	AM2_GRID_TB	31030	
5	Standard Deviation (V)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV_STD	34140	
6	Standard Deviation (H)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH_STD	34150	
7	Standard Deviation (V&H)	signed int	float	AMTK_get_GridFloat	—	AM2_GRID_TB_STD	34160	
8	Average Number (V)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBV_ANUM	34170	
9	Average Number (H)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBH_ANUM	34180	
10	Average Number (V&H)	signed int	signed int	AMTK_get_GridInt	—	AM2_GRID_TB_ANUM	34190	
11	Total Number (V)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBV_TNUM	34200	
12	Total Number (H)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBH_TNUM	34210	
13	Total Number (V&H)	signed int	signed int	AMTK_get_GridInt	—	AM2_GRID_TB_TNUM	34220	

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
EQR, Physical amount								
1	ProductMeta Data	-					-	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	Refer to Note 1 and 6.1.1.7
3						AM2_GRID_GEO2	31320	
4						AM2_GRID_GEO3	31330	
5						AM2_GRID_GEOA	31340	
6	Standard Deviation	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1_STD	34510	Refer to Note 1 and 6.1.1.7
7						AM2_GRID_GEO2_STD	34520	
8						AM2_GRID_GEO3_STD	34530	
9						AM2_GRID_GEOA_STD	34540	
10	Average Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_ANUM	34550	Refer to Note 1 and 6.1.1.7
11						AM2_GRID_GEO2_ANUM	34560	
12						AM2_GRID_GEO3_ANUM	34570	
13						AM2_GRID_GEOA_ANUM	34580	
14	Total Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_TNUM	34590	Refer to Note 1 and 6.1.1.7
15						AM2_GRID_GEO2_TNUM	34600	
16						AM2_GRID_GEO3_TNUM	34610	
17						AM2_GRID_GEOA_TNUM	34620	

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
PS The north, Sea Ice Concentration								
1	ProductMeta Data	-					-	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	Refer to Note 1 and 6.1.1.7
3						AM2_GRID_GEO2	31320	
4						AM2_GRID_GEO3	31330	
5						AM2_GRID_GEOA	31340	
6	Standard Deviation	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1_STD	34510	Refer to Note 1 and 6.1.1.7
7						AM2_GRID_GEO2_STD	34520	
8						AM2_GRID_GEO3_STD	34530	
9						AM2_GRID_GEOA_STD	34540	
10	Average Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_ANUM	34550	Refer to Note 1 and 6.1.1.7
11						AM2_GRID_GEO2_ANUM	34560	
12						AM2_GRID_GEO3_ANUM	34570	
13						AM2_GRID_GEOA_ANUM	34580	
14	Total Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_TNUM	34590	Refer to Note 1 and 6.1.1.7
15						AM2_GRID_GEO2_TNUM	34600	
16						AM2_GRID_GEO3_TNUM	34610	
17						AM2_GRID_GEOA_TNUM	34620	

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
PS The southern, Sea Ice Concentration								
1	ProductMeta Data	-					-	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	Refer to Note 1 and 6.1.1.7
3						AM2_GRID_GEO2	31320	
4						AM2_GRID_GEO3	31330	
5						AM2_GRID_GEOA	31340	
6	Standard Deviation	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1_STD	34510	Refer to Note 1 and 6.1.1.7
7						AM2_GRID_GEO2_STD	34520	
8						AM2_GRID_GEO3_STD	34530	
9						AM2_GRID_GEOA_STD	34540	
10	Average Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_ANUM	34550	Refer to Note 1 and 6.1.1.7
11						AM2_GRID_GEO2_ANUM	34560	
12						AM2_GRID_GEO3_ANUM	34570	
13						AM2_GRID_GEOA_ANUM	34580	
14	Total Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_TNUM	34590	Refer to Note 1 and 6.1.1.7
15						AM2_GRID_GEO2_TNUM	34600	
16						AM2_GRID_GEO3_TNUM	34610	
17						AM2_GRID_GEOA_TNUM	34620	

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
PS The south, Sea Ice Concentration								
1	ProductMeta Data	-					-	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	Refer to Note 1 and 6.1.1.7
3						AM2_GRID_GEO2	31320	
4						AM2_GRID_GEO3	31330	
5						AM2_GRID_GEOA	31340	
6	Standard Deviation	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1_STD	34510	Refer to Note 1 and 6.1.1.7
7						AM2_GRID_GEO2_STD	34520	
8						AM2_GRID_GEO3_STD	34530	
9						AM2_GRID_GEOA_STD	34540	
10	Average Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_ANUM	34550	Refer to Note 1 and 6.1.1.7
11						AM2_GRID_GEO2_ANUM	34560	
12						AM2_GRID_GEO3_ANUM	34570	
13						AM2_GRID_GEOA_ANUM	34580	
14	Total Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_TNUM	34590	Refer to Note 1 and 6.1.1.7
15						AM2_GRID_GEO2_TNUM	34600	
16						AM2_GRID_GEO3_TNUM	34610	
17						AM2_GRID_GEOA_TNUM	34620	

Monthly (Low resolution)

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
EQR, Brightness temperature								
1	ProductMeta Data	-					-	
2	Brightness Temperature (V)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV	31010	
3	Brightness Temperature (H)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH	31020	
4	Brightness Temperature (V&H)	unsigned int	float	AMTK_get_GridFloat	—	AM2_GRID_TB	31030	
5	Standard Deviation (V)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV_STD	34140	
6	Standard Deviation (H)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH_STD	34150	
7	Standard Deviation (V&H)	signed int	float	AMTK_get_GridFloat	—	AM2_GRID_TB_STD	34160	
8	Average Number (V)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBV_ANUM	34170	
9	Average Number (H)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBH_ANUM	34180	
10	Average Number (V&H)	signed int	signed int	AMTK_get_GridInt	—	AM2_GRID_TB_ANUM	34190	
11	Total Number (V)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBV_TNUM	34200	
12	Total Number (H)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBH_TNUM	34210	
13	Total Number (V&H)	signed int	signed int	AMTK_get_GridInt	—	AM2_GRID_TB_TNUM	34220	

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
PS The North , Brightness temperature								
1	ProductMeta Data	-					-	
2	Brightness Temperature (V)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV	31010	
3	Brightness Temperature (H)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH	31020	
4	Brightness Temperature (V&H)	unsigned int	float	AMTK_get_GridFloat	—	AM2_GRID_TB	31030	
5	Standard Deviation (V)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV_STD	34140	
6	Standard Deviation (H)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH_STD	34150	
7	Standard Deviation (V&H)	signed int	float	AMTK_get_GridFloat	—	AM2_GRID_TB_STD	34160	
8	Average Number (V)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBV_ANUM	34170	
9	Average Number (H)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBH_ANUM	34180	
10	Average Number (V&H)	signed int	signed int	AMTK_get_GridInt	—	AM2_GRID_TB_ANUM	34190	
11	Total Number (V)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBV_TNUM	34200	
12	Total Number (H)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBH_TNUM	34210	
13	Total Number (V&H)	signed int	signed int	AMTK_get_GridInt	—	AM2_GRID_TB_TNUM	34220	

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
PS The south, Brightness temeperature								
1	ProductMeta Data	-					-	
2	Brightness Temperature (V)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV	31010	
3	Brightness Temperature (H)	unsigned int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH	31020	
4	Brightness Temperature (V&H)	unsigned int	float	AMTK_get_GridFloat	—	AM2_GRID_TB	31030	
5	Standard Deviation (V)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBV_STD	34140	
6	Standard Deviation (H)	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_TBH_STD	34150	
7	Standard Deviation (V&H)	signed int	float	AMTK_get_GridFloat	—	AM2_GRID_TB_STD	34160	
8	Average Number (V)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBV_ANUM	34170	
9	Average Number (H)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBH_ANUM	34180	
10	Average Number (V&H)	signed int	signed int	AMTK_get_GridInt	—	AM2_GRID_TB_ANUM	34190	
11	Total Number (V)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBV_TNUM	34200	
12	Total Number (H)	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TBH_TNUM	34210	
13	Total Number (V&H)	signed int	signed int	AMTK_get_GridInt	—	AM2_GRID_TB_TNUM	34220	

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
EQR, Physical quantity								
1	ProductMeta Data	-					-	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	Refer to Note1 and 6.1.1.7
3						AM2_GRID_GEO2	31320	
4						AM2_GRID_GEO3	31330	
5						AM2_GRID_GEOA	31340	
6	Standard Deviation	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1_STD	34510	Refer to Note1 and 6.1.1.7
7						AM2_GRID_GEO2_STD	34520	
8						AM2_GRID_GEO3_STD	34530	
9						AM2_GRID_GEOA_STD	34540	
10	Average Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_ANUM	34550	Refer to Note1 and 6.1.1.7
11						AM2_GRID_GEO2_ANUM	34560	
12						AM2_GRID_GEO3_ANUM	34570	
13						AM2_GRID_GEOA_ANUM	34580	
14	Total Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_TNUM	34590	Refer to Note1 and 6.1.1.7
15						AM2_GRID_GEO2_TNUM	34600	
16						AM2_GRID_GEO3_TNUM	34610	
17						AM2_GRID_GEOA_TNUM	34620	

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
PS The North Sea Ice Concentration								
1	ProductMeta Data	-					-	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	Refer to 6.1.1.7
3						AM2_GRID_GEO2	31320	
4						AM2_GRID_GEO3	31330	
5						AM2_GRID_GEOA	31340	
6	Standard Deviation	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1_STD	34510	Refer to 6.1.1.7
7						AM2_GRID_GEO2_STD	34520	
8						AM2_GRID_GEO3_STD	34530	
9						AM2_GRID_GEOA_STD	34540	
10	Average Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_ANUM	34550	Refer to 6.1.1.7
11						AM2_GRID_GEO2_ANUM	34560	
12						AM2_GRID_GEO3_ANUM	34570	
13						AM2_GRID_GEOA_ANUM	34580	
14	Total Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_TNUM	34590	Refer to 6.1.1.7
15						AM2_GRID_GEO2_TNUM	34600	
16						AM2_GRID_GEO3_TNUM	34610	
17						AM2_GRID_GEOA_TNUM	34620	

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
PS The South, Sea Ice Concentration								
1	ProductMeta Data	-					-	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	Refer to 6.1.1.7
3						AM2_GRID_GEO2	31320	
4						AM2_GRID_GEO3	31330	
5						AM2_GRID_GEOA	31340	
6	Standard Deviation	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1_STD	34510	Refer to 6.1.1.7
7						AM2_GRID_GEO2_STD	34520	
8						AM2_GRID_GEO3_STD	34530	
9						AM2_GRID_GEOA_STD	34540	
10	Average Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_ANUM	34550	Refer to 6.1.1.7
11						AM2_GRID_GEO2_ANUM	34560	
12						AM2_GRID_GEO3_ANUM	34570	
13						AM2_GRID_GEOA_ANUM	34580	
14	Total Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_TNUM	34590	Refer to 6.1.1.7
15						AM2_GRID_GEO2_TNUM	34600	
16						AM2_GRID_GEO3_TNUM	34610	
17						AM2_GRID_GEOA_TNUM	34620	

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
PS The south, Sea Ice Concentration								
1	ProductMeta Data	-					-	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	Refer to 6.1.1.7
3						AM2_GRID_GEO2	31320	
4						AM2_GRID_GEO3	31330	
5						AM2_GRID_GEOA	31340	
6	Standard Deviation	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1_STD	34510	Refer to 6.1.1.7
7						AM2_GRID_GEO2_STD	34520	
8						AM2_GRID_GEO3_STD	34530	
9						AM2_GRID_GEOA_STD	34540	
10	Average Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_ANUM	34550	Refer to 6.1.1.7
11						AM2_GRID_GEO2_ANUM	34560	
12						AM2_GRID_GEO3_ANUM	34570	
13						AM2_GRID_GEOA_ANUM	34580	
14	Total Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_TNUM		Refer to 6.1.1.7
15						AM2_GRID_GEO2_TNUM		
16						AM2_GRID_GEO3_TNUM	34610	
17						AM2_GRID_GEOA_TNUM	34620	

PS The north, Snow Depth

1	ProductMeta Data	-					-	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	Refer to 6.1.1.7
3						AM2_GRID_GEO2	31320	
4						AM2_GRID_GEO3	31330	
5						AM2_GRID_GEOA	31340	
6	Standard Deviation	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1_STD	34510	Refer to 6.1.1.7
7						AM2_GRID_GEO2_STD	34520	
8						AM2_GRID_GEO3_STD	34530	
9						AM2_GRID_GEOA_STD	34540	
10	Average Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_ANUM	34550	Refer to 6.1.1.7
11						AM2_GRID_GEO2_ANUM	34560	
12						AM2_GRID_GEO3_ANUM	34570	
13						AM2_GRID_GEOA_ANUM	34580	
14	Total Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_TNUM	34590	Refer to 6.1.1.7
15						AM2_GRID_GEO2_TNUM	34600	
16						AM2_GRID_GEO3_TNUM	34610	
17						AM2_GRID_GEOA_TNUM	34620	

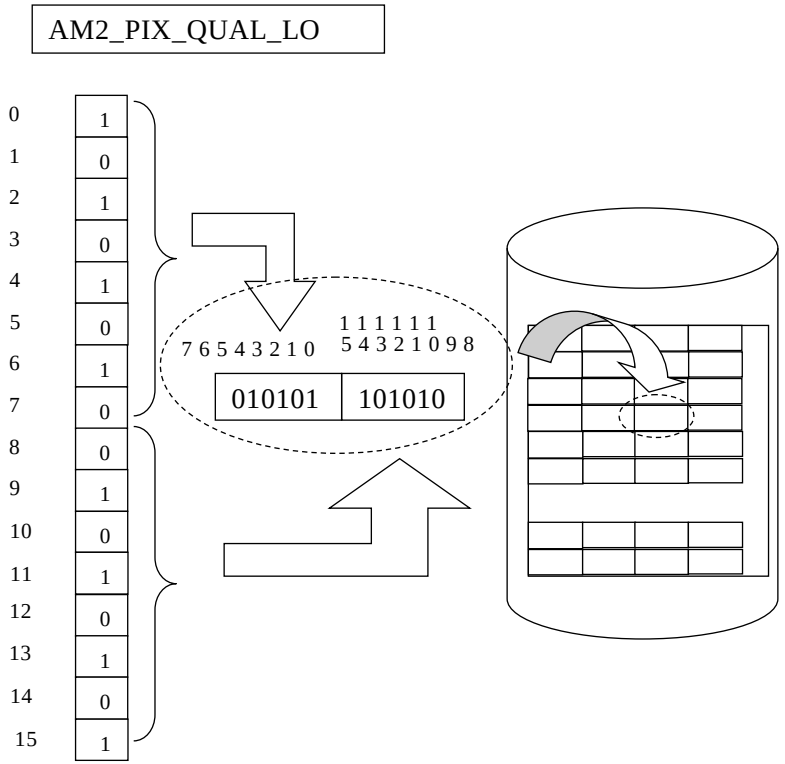
6.1.1.6 LATLON function

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
1	LatitudeLongitude (L1A/L1B 6.9GHz)	float	AM2_COMMON_LATLON	AMTK_getLatLon	—	AM2_LATLON_06	40010	Return the each channel lat/long-itude calculated by the each registration from 89A point.
2	LatitudeLongitude (L1A/L1B 7.3GHz)	float	AM2_COMMON_LATLON	AMTK_getLatLon	—	AM2_LATLON_07	40020	
3	LatitudeLongitude (L1A/L1B 10.7GHz)	float	AM2_COMMON_LATLON	AMTK_getLatLon	—	AM2_LATLON_10	40030	
4	LatitudeLongitude (L1A/L1B 18.7GHz)	float	AM2_COMMON_LATLON	AMTK_getLatLon	—	AM2_LATLON_18	40040	
5	LatitudeLongitude (L1A/L1B 23.8GHz)	float	AM2_COMMON_LATLON	AMTK_getLatLon	—	AM2_LATLON_23	40050	
6	LatitudeLongitude (L1A/L1B 36.5GHz)	float	AM2_COMMON_LATLON	AMTK_getLatLon	—	AM2_LATLON_36	40060	
7	LatitudeLongitude (L1A/L1B 89.0GHz-A)	float	AM2_COMMON_LATLON	AMTK_getLatLon	AMTK_setLatLon	AM2_LATLON_89A	40070	HDF hangar
8	LatitudeLongitude (L1A/L1B 89.0GHz-B)	float	AM2_COMMON_LATLON	AMTK_getLatLon	AMTK_setLatLon	AM2_LATLON_89B	40080	HDF hangar
9	LatitudeLongitude (L1A/L1B Low Mean)	float	AM2_COMMON_LATLON	AMTK_getLatLon	—	AM2_LATLON_LO	40090	Return the average value of the lat/long-itude of Low(06-36GHz)
10	LatitudeLongitude (L1R 89A)	float	AM2_COMMON_LATLON	AMTK_getLatLon	AMTK_setLatLon	AM2_LATLON_RS_89A	40100	Return the lat/long-itude.
11	LatitudeLongitude (L1R 89B)	float	AM2_COMMON_LATLON	AMTK_getLatLon	AMTK_setLatLon	AM2_LATLON_RS_89B	40110	Return the lat/long-itude.
12	LatitudeLongitude (L1R resampling)	float	AM2_COMMON_LATLON	AMTK_getLatLon	—	AM2_LATLON_RS_LO	40120	Return the resampling point even number of AM2_LATLON_RS_89A (start 1)
13	LatitudeLongitude (L2 Low-Res)	float	AM2_COMMON_LATLON	AMTK_getLatLon	AMTK_setLatLon	AM2_LATLON_L2_LO	40130	Return the lat/long-itude of L2.
14	LatitudeLongitude (L2 High-Res 89A)	float	AM2_COMMON_LATLON	AMTK_getLatLon	AMTK_setLatLon	AM2_LATLON_L2_89A	40140	
15	LatitudeLongitude (L2 High-Res 89B)	float	AM2_COMMON_LATLON	AMTK_getLatLon	AMTK_setLatLon	AM2_LATLON_L2_89B	40150	
16	LatitudeLongitude (L3 daily/monthly High-Res EQR)	float	AM2_COMMON_LATLON	AMTK_getLatLon	—	AM2_GRID_LATLON	40160	Return the lat/ling-itude from

Data name	Type(HDF)	Type	Access function	Access label		Access ID	Remark	Data name
				In	Out			
17	LatitudeLongitude (L3 daily/monthly High-Res PS North)	float	AM2_COMMON_LATLON	AMTK_getLatLon	—	AM2_GRID_LATLON	40160	HDF metadata. Specifying the Start/end number is invalid.
18	LatitudeLongitude (L3 daily/monthly High-Res PS South)	float	AM2_COMMON_LATLON	AMTK_getLatLon	—	AM2_GRID_LATLON	40160	
19	LatitudeLongitude (L3 daily/monthly Low-Res EQR)	float	AM2_COMMON_LATLON	AMTK_getLatLon	—	AM2_GRID_LATLON	40160	
20	LatitudeLongitude (L3 daily/monthly Low-Res PS North)	float	AM2_COMMON_LATLON	AMTK_getLatLon	—	AM2_GRID_LATLON	40160	
21	LatitudeLongitude (L3 daily/monthly Low-Res PS South)	float	AM2_COMMON_LATLON	AMTK_getLatLon	—	AM2_GRID_LATLON	40160	

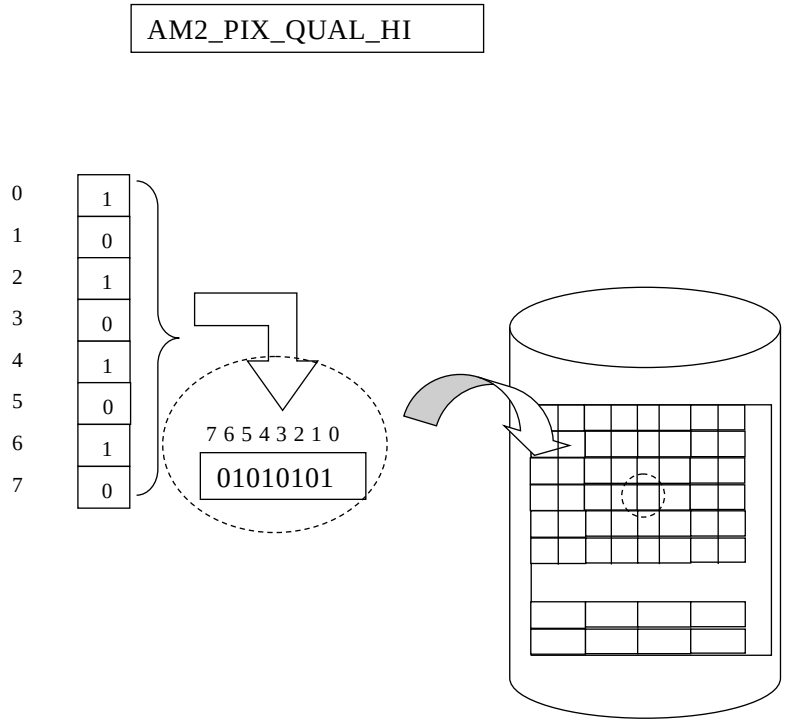
6.1.1.7 Paticular Dataset

(1) PIXEL DATA QUALITY(Access label：AM2_PIX_QUAL_LO ,AM2_PIX_QUAL_HI)
Plural unsigned char data are collected up and stored HDF.



Write: convert from 16bytes data to 2bytes.

Read: spread 2 bytes into 16 bytes data.



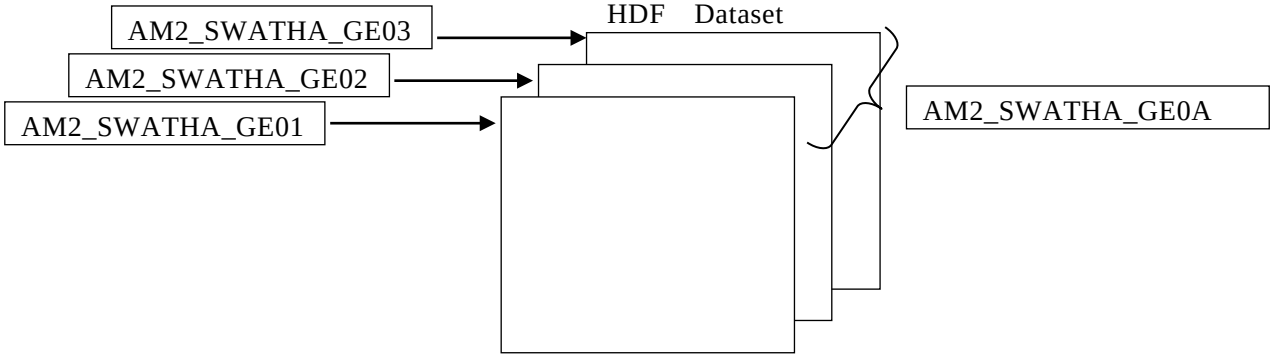
Write: convert from 8 bytes data to 1 bytes.

Read: spread 1 bytes into 8 bytes data.

(2) Give Specification for the layer structure (Access label : AM2_SWATHA_GEXX ,AM2_GRID_GEXX)

Note the differences of the data set of the layer structure and the designated of the access label at the time of generation and the writing.

Show the a concept in the following figures.



Generate : Create the three-dimension data with specifying the access label for AM2_SWATHA_GE0A.

Read/Write : read/write to the two-dimension data with specifying the access label for AM2_SWATHA_GE01, GE02 and GE03.

The array elements are different as follows to let a scan number position support in Create and Read/Write.

AM2_SWATHA_GE0A : Scan number is appointed as a two-dimensional element.

AM2_SWATHA_GE01 ,GE02 ,GE03 : Scan number is appointed as a one-dimensional element.

6.1.2 Geophysical quantity dataset

6.1.2.1 3-dimension dataset

In AMTK, it classify 3-dimensional data set in two kinds of “nomal 3-dimensional data set” and “3-dimensinal data set a bout the geophysical quantity”. Then it offers a different input and output method each.

Both have each characteristic of “dimensions to express the number of the scan” when it in/out the size of data set. Table 6.1-1 shows each input and output method.

Table 6.1-1 List of input and output methods of the 3-dimensional data set

No.	Type	Specification of the input and output
1	nomal 3-dimensional data set	When It inputs and outputs the size of the data set with AMTK_setDimSize() or AMTK_getDimSize(), the number of the scan of the dimension size assumes it the second dimension.

		Ex.)
		The size of Hot Load Count 6 to 36: number of channel * number of scan * number of pixel
		dimsize[0] = 12
		dimsize[1] = 2000 // number of scan is at the second dimension.
		dimsize[2] = 16

2	3-dimensinal data set about the ge ophysical quantity (Data set to access each layer hav ing 1-3 levels two-dimensional dat a individually)	When it input and output the size of the data set with AMTK_setDimSize() or AMTK_getDimSize(), the number of the scan of the dimension size assumes it the 1st dimension and the number of the layer assumes it the 3rd dimension.

		Ex.)
		The size of Geopysical Data: number of scan * number of pixel * number of layer
		dimsize[0] = 2000 // number of scan is at the 1st dimension.
		dimsize[1] = 243
		dimsize[2] = 3 // number of layer is at the third dimension.

6.1.2.1.1 Original 3-dimension dataset list

○L1

No.	Data set name	HDF storage type	In/put type	Input function	Output function	Access label	Access ID
L1A, L1B							
1	Hot Load Count 6 to 36	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_HTS_LO	10240
2	Hot Load Count 89	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_HTS_HI	10250
3	Cold Sky Mirror Count 6 to 36	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_CSM_LO	10260
4	Cold Sky Mirror Count 89	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_CSM_HI	10270
5	Land_Ocean Flag 6 to 36	unsigned char	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_LOF_LO	10340
6	Land_Ocean Flag 89	unsigned char	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_LOF_HI	10350
7	Pixel Data Quality 6 to 36	binary (2byte) = unsigned char*2	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_LO	10470
8	Pixel Data Quality 89	binary (1byte) = unsigned char*1	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_HI	10480
9	Interpolation Flag 6 to 36	binary (1byte) = unsigned char*1	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_INTPL_LO	10490
10	Interpolation Flag 89	binary (1byte) = unsigned char*1	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_INTPL_HI	10500
L1R							
11	Land_Ocean Flag 6 to 36	unsigned char	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_LOF_RES_LO	12560
12	Land_Ocean Flag 89	unsigned char	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_LOF_RES_HI	12570

13	Pixel Data Quality 6 to 36	binary (2byte) = unsigned char*2	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_LO	10470
14	Pixel Data Quality 89	binary (1byte) = unsigned char*1	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_HI	10480

○L2

No.	Data set name	HDF storage type	In/put type	Input function	Output function	Access label	Access ID
Low resolution							
1	Pixel Data Quality	binary (1byte) = unsigned char*1	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL	21070
High resolution							
2	Pixel Data Quality for 89A	binary (1byte) = unsigned char*1	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_A	22110
3	Pixel Data Quality for 89B	binary (1byte) = unsigned char*1	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_B	22120

6.1.2.1.2 3-dimension dataset of Geophysical quantity list

・AM2_*A of the access label inputs and outputs it as 3-dimensional data set (Scan * Pixel * Layer size).

・AM2_*1 - *3 of the access label inputs and outputs 2-dimensions of data of the layer of 1-3 as data set (Scan * Pixel siz e).

○L2

No.	Data set name	HDF storage type	In/put type	Input function	Output function	Access label	Access ID
1	Geophysical Data	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_SWATH_GEO1	21030
2						AM2_SWATH_GEO2	21040
3						AM2_SWATH_GEO3	21050
4						AM2_SWATH_GEOA	21060
5	Geophysical Data for 89A	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_SWATHA_GEO1	22030
6						AM2_SWATHA_GEO2	22040
7						AM2_SWATHA_GEO3	22050
8						AM2_SWATHA_GEOA	22060
9	Geophysical Data for 89B	signed int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_SWATHB_GEO1	22070
10						AM2_SWATHB_GEO2	22080
11						AM2_SWATHB_GEO3	22090
12						AM2_SWATHB_GEOA	22100

○L3

No.	Data set name	HDF storage type	In/put type	Input function	Output function	Access label	Access ID
1	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310
2						AM2_GRID_GEO2	31320
3						AM2_GRID_GEO3	31330
4						AM2_GRID_GEOA	31340
5	Standard Deviation	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1_STD	34510
6						AM2_GRID_GEO2_STD	34520
7						AM2_GRID_GEO3_STD	34530
8						AM2_GRID_GEOA_STD	34540
9	Average Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_ANUM	34550
10						AM2_GRID_GEO2_ANUM	34560
11						AM2_GRID_GEO3_ANUM	34570
12						AM2_GRID_GEOA_ANUM	34580
13	Total Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GEO1_TNUM	34590
14						AM2_GRID_GEO2_TNUM	34600
15						AM2_GRID_GEO3_TNUM	34610
16						AM2_GRID_GEOA_TNUM	34620

6.1.2.1.3 A dimension to express the number of the scan of the 3-dimensional data set

The normal 3-dimensional data set handles the secound dimension with the number of the scan. On the other hand, t he 3-dimensional data set about the quantity of physics handles the first dimension with the number of the scan. I show the difference of the dimension to express the number of the scan in the dimension s ize of both to

Fig. 6-1.

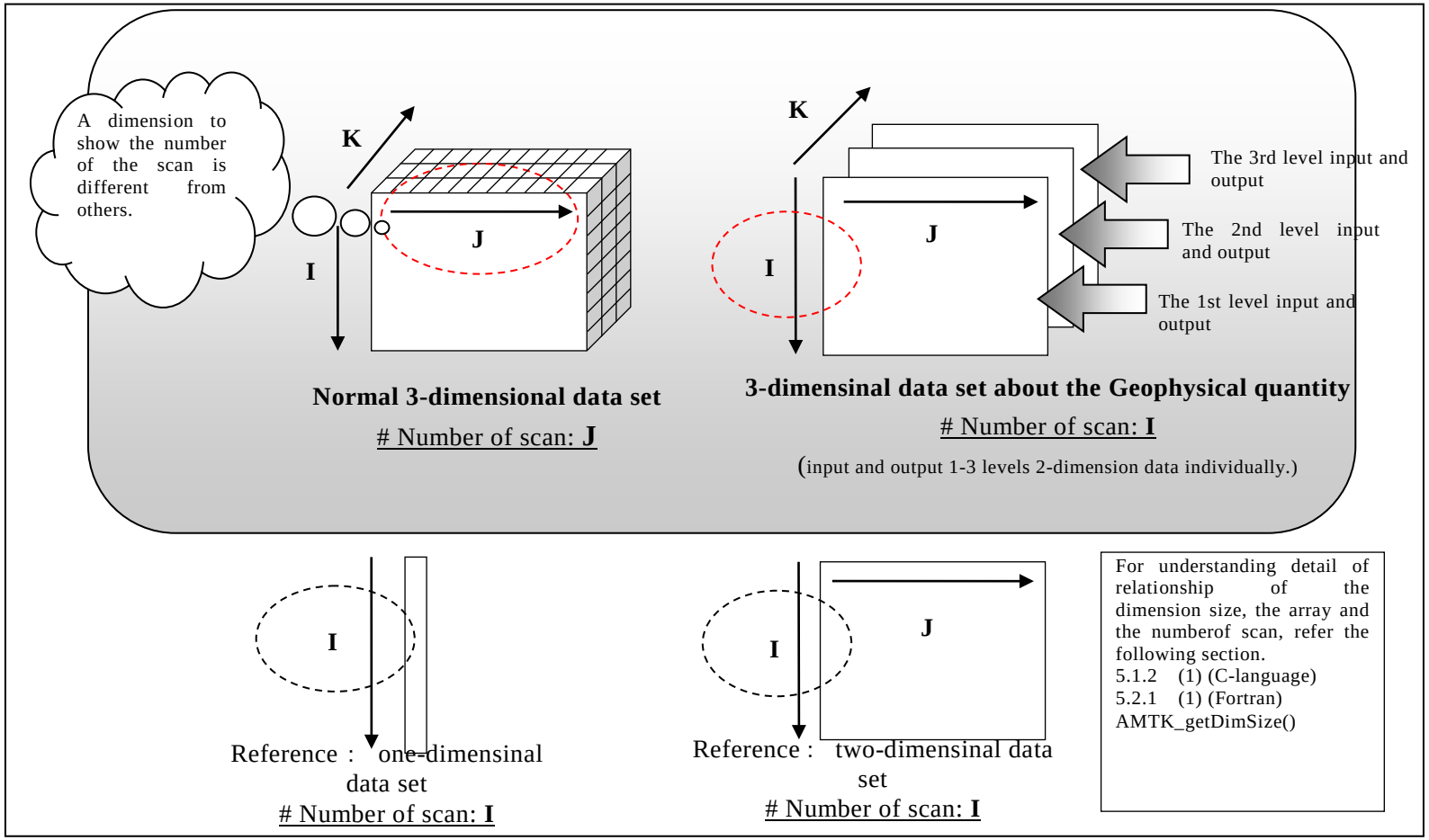


Fig. 6.1-1 The difference of the dimension to express the number of the scan in the D men John size

The Geophysical Data dataset of L2 or L3, Standard Deviation of L3, Average Number and Total number dataset are 3-dimensinal dataset about the geophysical quantity. Then its structure has the 1st level – the 3rd levels 2-dimensional data.

AMTK offers the 2 kinds of access label. The one inputs and outputs all 3-dimensins of layers by a lump as data. The other inputs and outputs 2-dimension data of 1st level, the 2nd level and the 3rd level individually.

The image is shown in Fig. 6-2

・Lump access label：AM2_* _GEOA* (3-dimensional data access for all layers)

For AMTK_setDimSize() , AMTK_getDimSize()

・individually access label：AM2_* _GEO1* ,AM2_* _GEO2* ,AM2_* _GEO3* (2-dimensional data access for each layer)

For AMTK_set_Swath*(), AMTK_get_Swath*(), AMTK_set_Grid*(), AMTK_get_Grid*()

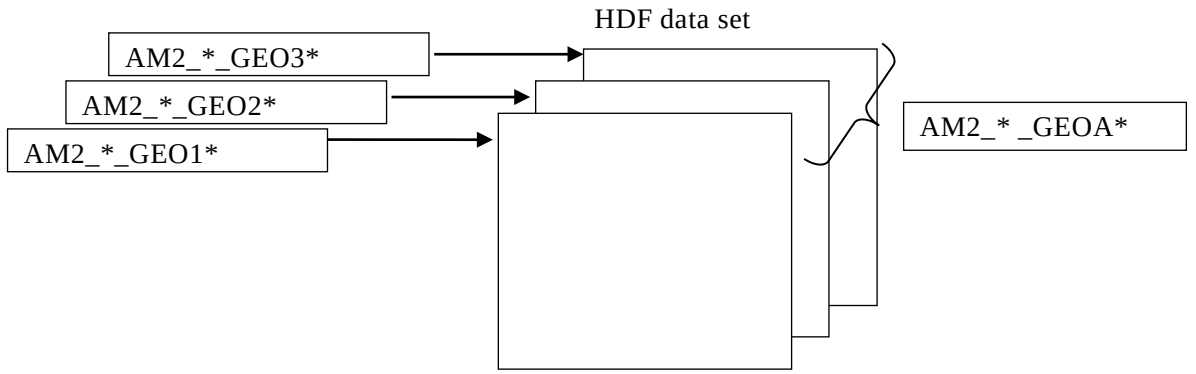


Fig. 6.1-2 Two dimensions data individual access label and 3-dimensional data lump access label

For a concrete input/output example of a normal 3-dimensinal data set or a 3-dimensinal data set about the geophysical quantity, refer the input/output sample program of L3.

In the following, the part of the input/output sample program of L2 is explained.

(1) Output sample of a normal 3-dimensional data set

sample3_make_L2Lproduct.c : Pixel Data Quality dataset (part extracted)

```
...

/** Number of Geophysical Data. */
#define GEO_DATA_LAYER_NUM      (3)
...
 /* Pixel Data Quality
   * (type size * Number of Geophysical Data[1...3] * Numbef of scans * 243) */
 p_dataset->p_pixel_quality = (unsigned char *) malloc(
                GEO_DATA_LAYER_NUM * scan_size * AM2_DEF_SNUM_LO);
 if (NULL == p_dataset->p_pixel_quality)
 {
                E_MSG("malloc() error.¥n");
                return RET_ERROR;
 }
...
 /* Pixel Data Quality */
 {
                dimsize[0] = GEO_DATA_LAYER_NUM; /* 1 ... 3 */
                dimsize[1] = scan_size;
                dimsize[2] = AM2_DEF_SNUM_LO;       /* binary (1byte) */
                ret = AMTK_setDimSize(file_id, AM2_PIX_QUAL, dimsize);
                if (0 > ret)
                {
                                E_MSG("AMTK_setDimSize() error.[%d]¥n", ret);
                                terminate(file_id, &dataset);
                                exit(EXIT_FAILURE);
                }

                ret = AMTK_set_SwathUChar(file_id, dataset.p_pixel_quality,
                                scan_start, scan_end, AM2_PIX_QUAL);
                if (0 > ret)
                {
                                E_MSG("AMTK_set_SwathUChar() error.[%d]¥n", ret);
                                terminate(file_id, &dataset);
                                exit(EXIT_FAILURE);
                }
 }
```

●declaration
In this exsample, the number of geophysical quantity (layer) of Pixel Data Quality data set to output is 3.

●memory allocation
It allocates the area of the data set to output.

Memory area : number of layer * number of scan * number of pixel
It allocate a memory area using malloc().

●create the dataset
It set dimension size to output.
At this time, it set the number of scan in the 2nd layer.

1st dimension: number of Layer
2nd dimension: number of scan
3rd dimension: number of pixel

It outputs the 3-dimensional dataset of (number of layer * number of scan * number of pixel) size using AM2_PIX_QUALaccess label.

●write data
It set the value of data set to output

(2) Output sample of a3-dimensinal data set about the Geophysical quantity

sample3_make_L2Lproduct.c :Geophysical Data dataset output(part extracted)

```
...

/** Number of Geophysical Data. */
#define GEO_DATA_LAYER_NUM      (3)
...
 /* Access label: Geophysical Data */
 const int geo_data_label[] = {AM2_SWATH_GEO1, AM2_SWATH_GEO2, AM2_SWATH_GEO3};
...
 /* Geophysical Data (Layer: 1...3)
   * (type size * Number of scans * 243) */
 for (i = 0; i < GEO_DATA_LAYER_NUM; i++)
 {
        p_dataset->p_geo_data[i] = (float *) malloc(sizeof(float) * scan_size
                * AM2_DEF_SNUM_LO);
        if (NULL == p_dataset->p_geo_data[i])
        {
                E_MSG("malloc() error.¥n");
                return RET_ERROR;
        }
 }
...
 /* Geophysical Data (Layer: 1...3) */
 {
        dimsize[0] = scan_size;
        dimsize[1] = AM2_DEF_SNUM_LO;
        dimsize[2] = GEO_DATA_LAYER_NUM;    /* 1 ... 3 */

        /* Layer: ALL -> AM2_SWATH_GEOA */
        ret = AMTK_setDimSize(file_id, AM2_SWATH_GEOA, dimsize);
        if (0 > ret)
        {
                E_MSG("AMTK_setDimSize() error.[%d]¥n", ret);
                terminate(file_id, &dataset);
                exit(EXIT_FAILURE);
        }

        /* Layer: 1...3 -> AM2_SWATH_GEO1 ... AM2_SWATH_GEO3 */
        for (i = 0; i < GEO_DATA_LAYER_NUM; i++)
        {
                ret = AMTK_set_SwathFloat(file_id, dataset.p_geo_data[i], scan_start,
                        scan_end, geo_data_label[i]);
                if (0 > ret)
                {
                        E_MSG("AMTK_set_SwathFloat() error.[%d]¥n", ret);
```

●declaration
In this exsample, the number of geophysical quantity (layer) of Geophysical Data data set to output is 3.

●memory allocation
It allocates the area of the data set to output.

Memory area : number of scan * number of pixel * 3 layers
It allocate a memory area using malloc().

●create the dataset
It set dimension size to output.
At this time, it set the number of scan in the 1st layer.

1st dimension: **: number of scan**
2nd dimension: number of pixel
3rd dimension: number of Layer

It outputs the 3-dimensional dataset of (number of layer * number of scan * number of pixel) size using AM2_SWATH_GEOA access label.

●write data
It set the value of data set to output

It outputs the value of the 2-dimensional data set for 3 levels in total using the access label of the 1st layer to 3rd layer of Geophysical Data dataset.

```
                terminate(file_id, &dataset);
                exit(EXIT_FAILURE);
            }
        }
    }
    ...
```

6.1.2.2 Pixel Data Quality dataset stored method

When it input/output the Pixel Data Quality dataset using AMTK, C-language uses unsigned chartype array and Fortran uses charactertype array.

The value to set is stored amay by bit of each element.

Pixel Data Quality 6 to 36 of L1 makes sense by 16bits unit (array size = 2) . And Pixel Data Quality 89 of L1 makes sense by 8bits unit (arraysize = 1).

The stored image is shown in Fig. 6-3.

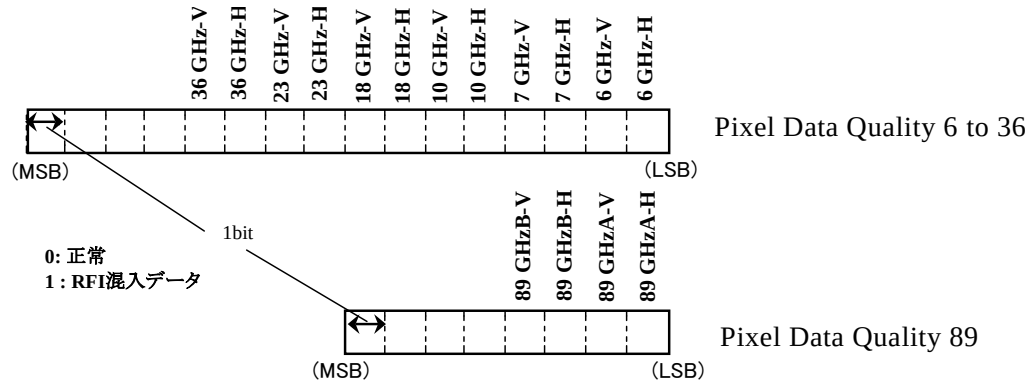


Fig. 6.1-3 Stored image (Pixel Data Quality 6 to 36, 89 of L1)

It input/output to HDF file by 1byte unit.

The image of the data set and the array of Pixel Data Quality 6 to 36 and Pixel Data Quality 89 is shown in Fig. 6-4

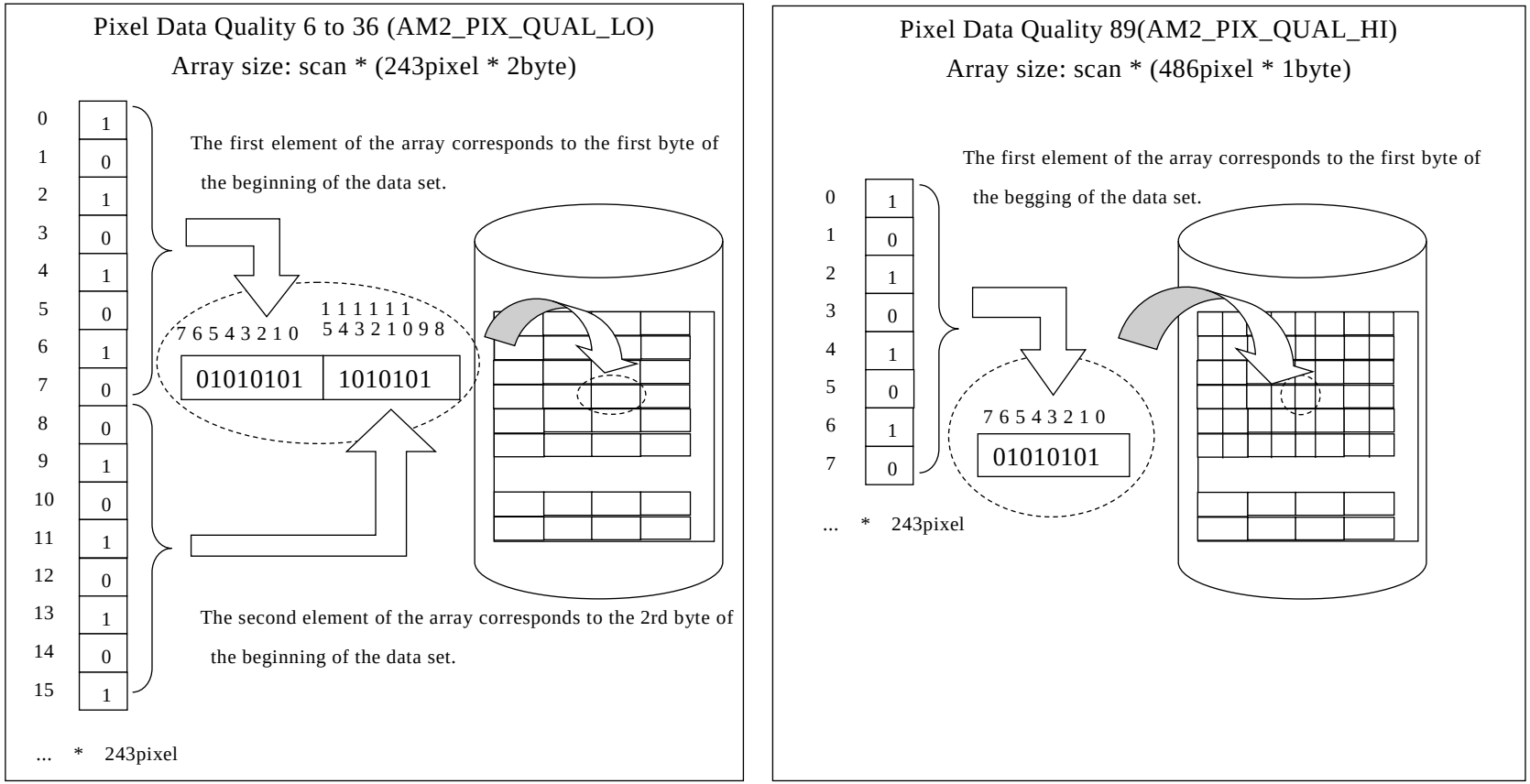


Fig. 6.1-4 Corresponding image (data sets and array for Pixel Data Quality 6 to 36, 89 of L1)

6.1.3 Product related iformation

	Name	Value	Description	Remark
Sample number	AM2_DEF_SNUM_LO	243	(AMSR-E A2LOW 196)	
	AM2_DEF_SNUM_HI	486	(AMSR-E A2HIGH 392)	
	AM2_DEF_L3H_EQ_X	3600	Number of pixels in X direction(longitude) for L3 High-res EQ	
	AM2_DEF_L3H_EQ_Y	1800	Number of pixels in Y direction(latitude) for L3 High-res EQ	
	AM2_DEF_L3L_EQ_X	1440	Number of pixels in X direction(longitude) for L3 Low-res EQ	
	AM2_DEF_L3L_EQ_Y	720	Number of pixels in Y direction(latitude) for L3 Low-res EQ	
	AM2_DEF_L3H_EQ_GSIZE	0.1	Grid size for L3 High-res EQ	
	AM2_DEF_L3L_EQ_GSIZE	0.25	Grid size for L3 Low-res EQ	
	AM2_DEF_L3H_PN1_X	760	Number of pixels in X direction for L3 High-res PN (TB,SIC)	
	AM2_DEF_L3H_PN1_Y	1120	Number of pixels in Y direction for L3 High-res PN (TB,SIC)	
	AM2_DEF_L3L_PN1_X	304	Number of pixels in Y direction for L3 Low-res PN (TB,SIC)	
	AM2_DEF_L3L_PN1_Y	448	Number of pixels in X direction for L3 Low-res PN (TB,SIC)	
	AM2_DEF_L3H_PN2_X	1078	Number of pixels in Y direction for L3 High-res PN (SND)	
	AM2_DEF_L3H_PN2_Y	1432	Number of pixels in X direction for L3 High-res PN (SND)	
	AM2_DEF_L3L_PN2_X	431	Number of pixels in Y direction for L3 Low-res PN (SND)	
	AM2_DEF_L3L_PN2_Y	573	Number of pixels in X direction for L3 Low-res PN (SND)	
	AM2_DEF_L3H_PS_X	790	Number of pixels in Y direction for L3 High-res PS	
	AM2_DEF_L3H_PS_Y	830	Number of pixels in X direction for L3 High-res PS	
	AM2_DEF_L3L_PS_X	316	Number of pixels in Y direction for L3 Low-res PS	
	AM2_DEF_L3L_PS_Y	332	Number of pixels in X direction for L3 Low-res PS	
Emendatory information	AM2_DEF_CAL_LO	16	CAL data sampling number(AMSR-E 16)	
	AM2_DEF_CAL_HI	32	CAL data sampling number(AMSR-E 32)	
Missing value	AM2_DEF_IMISS	-32768	signed integer loss value	
	AM2_DEF_UIMISS	65535	unsigned integer loss value	
	AM2_DEF_CMISS	255	character loss value	
	AM2_DEF_RMISS	-9999.99	real loss value	
HDF create mode	AM2_RW_MODE	1	AMTK_openH5_Write() : read/write mode	
	AM2_CREATE_MODE	2	AMTK_openH5_Write() : create mode	

6.2 L1, L2 ,L3 Common data

Show the common data structure as following.

Applied data		Data structure	Remark
	Scan time	AM2_COMMON_SCANTIME	
	Lat/Long-titude	AM2_COMMON_LATLON	
	Data quality	AMTK_SCAN_DATA_QUALITY	For L1

Scan time			
AM2_COMMON_SCANTIME			
Name	Type	Size	Description
tai93sec	double	1	The elapsed seond time from january 1st, 1993
year	short	1	Year
month	short	1	Month
day	short	1	Day
hour	short	1	Hour
minute	short	1	Minute
second	short	1	Second
ms	short	1	Mili second

Lat/Long-itude			
AM2_COMMON_LATLON			
Name	Type	Size	Description
lat	float	1	Latitude[deg]
lon	float	1	Longitude[deg]

Quality data			
AMTK_SCAN_DATA_QUALITY			
Name	Type	Size	Description
scan_quality	unsigned int[]	4	Quality for a Scan
calibration	float	64	Calibration Data
Spc_sps_error	unsigned int	1	SPC/SPS Error
Hts	unsigned int	16	HTS Temperature
parity	unsigned int[]	33	Parity Error Summary
quality_info	unsigned int[]	4	Quality Information for a scan
spare	unsigned int[]	6	Spare

7 Error Number ID

Show the error ID of AMTK functions as following.

All of ID are negative value and assigned to some categories.

-100～ Execution environment error

-200～ the error cause of the user program's bug.

-500～ the error cause of HDF file

-600～ the error cause of appricated data

Table 7-1 Error number ID

ID	Description	Remark
-100	Fail to opne the file	
-101	Fail to allocate memory	
-102	Fail to get evaluate variable	
-103	Fail to get leap second	
-109	Fail to get geophysical quantity file	
-201	Specify an invalid value(year of scan time)	After 1993
-202	Specify an invalid value(month of scan time)	Valid data: 1-12
-203	Specify an invalid value(day of scan time)	Valid data:1-31
-204	Specify an invalid value(hour of scan time)	Valid data:1-24
-205	Specify an invalid value(minute of scan time)	Valid data:0-59
-206	Specify an invalid value(second of scan time)	Valid data:0-60
-207	Specify an invalid value(mili second of scan time)	Valid data:0-999
-208	Specify an invalid value(mili second of scan time)	
-209	Specify an invalid value(elapsed second time)	
-210	No file name	
-211	A start scan number is a big value than an end scan number	
-213	HDF file ID is 0 or a minus value.	
-214	The index value of meta data is a minus value.	
-215	Appointed dimension size is unjust.	
-216	No specified wire data	
-217	Access label is 0.	
-218	Specify an invalid value(Access label is out of write object.)	
-219	Specify an invalid value(Access label is out of read object.)	
-220	0 size data acquires a memory area.	
-221	Specify an NULL address.	
-238	Invalid write mode for HDF file.	
-239	Unadapted data type for Access Label.	
-250	Fail to get three days average value responsible illigal geophysical quantity. "GeophysicalName" in metadata should be same in all input files.	reference: 5 API function Acquisition for 3days average data AMTK_get_Grid03D()
-251	Fail to get 3days average value so that "OrbitDirection" dosenot match a rule. It is necessary to set orbit direction in Descending in the case of the even number file in Ascending in the case of the odd number file in orbit direction.	eference: 5 API function Acquisition for 3days average data AMTK_get_Grid03D()
-252	Fail to get 3days average value so that "ProductionDateTime" dose not match a rule.	eference: 5 API function

ID	Description	Remark
	The date of "ProductionDateTime" is necessary to be same with an add number file and an even number file. And that, the filename1 is set as appointed day(ex. N), the filename2 is set as the before day of appointed day (ex. N-1) and the filename3 is set as the before day of filename2(ex. N-2).	Acquisition for 3days average data AMTK_get_Grid03D()
-253	Fail to get 3days average value so that a size of ophysical quantity dose not much a rule. It is necessary for a file acquiring average to be same as for ths size of "GeophysicalData".	eference: 5 API function Acquisition for 3days average data AMTK_get_Grid03D()
-501	Fail to open HDF file.	
-502	Fail to close HF file.	
-503	Fail to set dimension size.	
-504	Fail to access to data set.	
-505	Fail to open data set.	
-506	Fail to get space for data set.	
-507	Fail to get the number of array for data set.	
-508	Fail to get the size of dimension for data set.	
-509	Fail to create data set.	
-510	Fail to write to data set.	
-511	Fail to read from data set.	
-520	Fail to open the name of metadata.	
-521	Fail to fet the kind od data of metadata name.	
-522	Fail to read the value of metadata.	
-523	Fail to open with specified name of metadata.	
-524	Fail to create the name of metadata.	
-525	Fail to wirte the value of metadata.	
-526	Fail to get the class of data type.	
-527	Fail to copy the class of data type.	
-528	Fail to modify the size of data type.	
-529	Fail to get data type.	
-540	Fail to get the space of data type.	
-541	Fail to specify the point to read/writefor data set.	
-550	Fail to access data set so that the opening error ofmeta data “ProductName"	Reference : 6.1.1 HDF access lave
-551	Fail to access data set so that the internal error when it is reading metadata “ProductName"	Reference : 6.1.1 HDF access lave
-552	Fail to access data set so that the open error of meta data “OverlapScans”.	Reference : 6.1.1 HDF access lave
-553	Fail to access data set so that the internal error when it is reading metadata “OverlapScans".	Reference : 6.1.1 HDF access lave
-554	Fail to access data set so that the open error of meta data “GeophysicalName”.	Reference : 6.1.1 HDF access lave
-555	Fail to access data set so that the internal error when it is reading metadata “GeophysicalName ".	Reference : 6.1.1 HDF access lave
-556	Fail to access data set so that the open error of meta data “GranuleID”	Reference : 6.1.1 HDF access lave
-557	Fail to access data set so that the internal error when it is reading metadata “GranuleID ".	Reference : 6.1.1 HDF access lave
-558	Fail to access data set so that the open error of meta data “CoRegistrationParameterA1”	Reference : 6.1.1 HDF access lave
-559	Fail to access data set so that the internal error when it is reading metadata “CoRegistrationParameterA1".	Reference : 6.1.1 HDF access lave

ID	Description	Remark
-560	Fail to access data set so that the open error of meta data “CoRegistrationParameterA2”	Reference : 6.1.1 HDF access lave
-561	Fail to access data set so that the internal error when it is reading metadata “CoRegistrationParameterA2".	Reference : 6.1.1 HDF access lave
-562	Fail to access data set so that the open error of meta data “Projection”	Reference : 6.1.1 HDF access lave
-563	Fail to access data set so that the internal error when it is reading metadata “Projection ".	Reference : 6.1.1 HDF access lave
-564	Fail to access data set so that the open error of meta data “OrbitDirection”	Reference : 6.1.1 HDF access lave
-565	Fail to access data set so that the internal error when it is reading metadata “OrbitDirection ".	Reference : 6.1.1 HDF access lave
-566	Fail to access data set so that the open error of meta data “ProductionDateTime”	Reference : 6.1.1 HDF access lave
-567	Fail to access data set so that the internal error when it is reading metadata “ProductionDateTime ".	Reference : 6.1.1 HDF access lave
-601	Unadapted data type for read/write data.	
-602	Detect the outlier of maximam/minimamvalue for write data.	
9999	Write data is over the maximum value.	