GCOM-W1
AMSR2 プロダクト I/O ツールキット
取扱説明書

改訂履歴

版数	発行日	改訂ページ	改訂理由
初版	2010/03	―	―
1.11	2013/01/24	―	リリース版
1.12	2013/05/17	―	一部誤記を修正
1.13	2013/12/05	―	降水量の Scale factor を変更。 0.1→0.01
1.14	2015/03/26	―	欠損値のデータ定義を変更。 ・AM2_DEF_RMISS の定義値の修正。 −9999.00→−9999.99

目　次

1　HDF ライブラリと AMSR2 I/O ツールキット

AMSR2 I/O ツールキット（以降 AMTK と呼びます。）は、第一期水循環変動観測衛星（GCOM-W1）のデータである AMSR2 プロダクトデータを、C 言語や Fortran 言語のプログラムで利用するために提供されます。AMSR2 プロダクトデータは、レベル 1～3 に分類され、HDF (Hierarchical Data Format)というファイル形式に格納されます。AMTK は、HDF の AMSR2 プロダクトデータに容易にアクセスできるようにします。

1.1　HDF とは

AMSR2プロダクトデータは、すべてHDFのファイル形式に格納されています。

HDFは、NCSA(The National Center for Supercomputing Applications ： イリノイ大学)がユーザの計算機構成に依存せず、情報を利用できるように開発したファーマットです。HDFファイルは、HDF4形式とHDF5形式があり、HDF5は、HDF4の課題（データサイズに制限がある､複数種類のデータタイプなど)を全面的に見直して提供されています。AMSR2プロダクトでは、HDF5のファイル形式を適用し、AMTKは、HDF5のファイルにアクセスし、ユーザプログラムで指定された情報を取得（入力機能）または、設定（出力機能）します。

HDFファイルは、属性情報を持つAttribute部とプロダクト情報そのものをもつデータセット部に分けられます。AMSR2プロダクトのAttribute部には、メタデータが収容され、データセット部には、観測データ、緯度経度データなどプロダクトデータが収容されます。

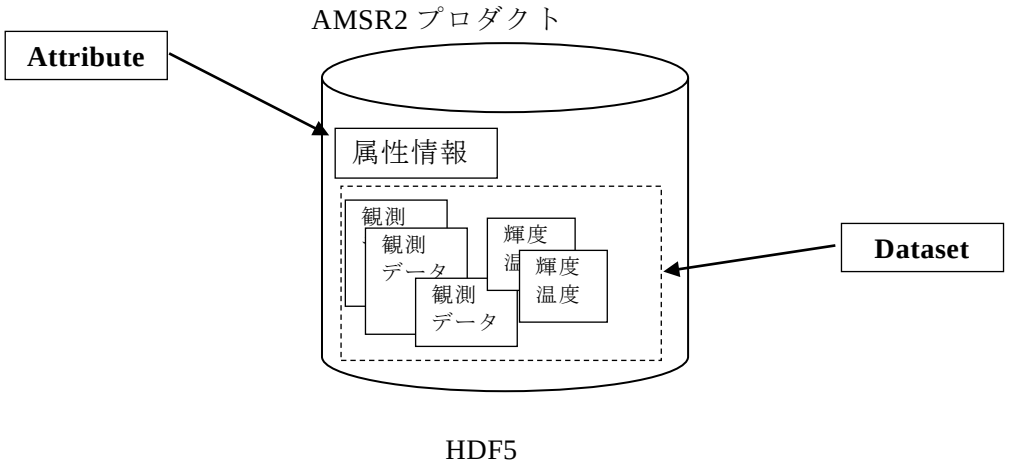


図 1-1 HDF ファイルの構成

1.2　AMTK とは

AMTKは、C言語やFortran言語のプログラム上で、AMSR2プロダクトデータ、DEMファイル、気象ファイルに容易にアクセスするため開発されたツールキットです。

AMTKは、AMSR2プロダクト情報の入出力関数群で構成されます。

HDF へのアクセスは、HDF ファイルをオープンし、このオープンにより返される識別子（hid_t）を指定して Attribute、各データセットの情報の入出力を行います。AMTK では、複数のデータセットを識別するため、HDF 識別番号（詳細は、6.1.1　章を参照してください。）を使用してデータセットを特定します。

(1) 動作確認環境

AMTK は、表 1.2-1に示す環境で動作確認を行いました。

表 1.2-2 動作確認環境

機種名	DELL4		Sun-Fire-V440	SGI Origin2000
OS	Red Hat Enterprise Linux Client release 5 (Tikanga)		SunOS 5.9	IRIX64 6.5
コンパイラ	C 言語	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		
メモリサイズ	12GB		8GB	1GB
HDF ライブラリ	HDF5-1.8.4-patch1 (※2)			

(※1)

g95 コンパイラは、G95 の Web サイトにて公開されている以下のバイナリを使用しました。

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

(※2)

Sun-Fire-V440 は HDF ライブラリをソースコードからコンパイルする必要があります。

SGI Origin2000 は HDF ライブラリのソースコードを一部改変してコンパイルする必要があります。

(インストール手順は、2.1　HDF5 ライブラリのインストールをご覧下さい)

(2) プラットフォーム

AMTK は、32 ビットマシン／64 ビットマシンの両方に対応します。

(3) 適用言語／使用コンパイラ

AMTK は、表 1.2-2 に示す言語、コンパイラで使用します。

表 1.2-3　適用言語／使用コンパイラ

OS	C 言語	Fortran 言語	
		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　HDF ライブラリと AMTK のインストール

AMSR2プロダクトデータは、HDF5ライブラリのRelease 1.8.4 Patch1を使用して作成されています。ここでは、HDF5ライブラリのRelease 1.8.4 Patch1のインストールについて説明します。

2.1　HDF5 ライブラリのインストール

HDF5ライブラリのインストール方法は、Release 1.8.4 Patch1のコンパイル済みのバイナリデータをインストールする方法と、HDF5ライブラリのソースコードをコンパイルする方法の2通りがあります。

対象のプラットフォームのバイナリが公開されていない場合や、バイナリを使用することで問題が発生する場合は、HDF5ライブラリのソースコードをコンパイルしインストールします。SGI環境に関しては、HDF5ライブラリのソースコードを一部改変した上でコンパイルする必要があります。いずれの場合も、下記の手順に従ってインストールしてください。

2.1.1　HDF5 ライブラリの入手

HDF グループのホームページから HDF5 ライブラリを入手します。HDF5 ライブラリ最新版のダウンロードのホームページを以下に示します。

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

2010 年 3 月時点でのダウンロードの対象を表に示します。ご使用になる計算機及び OS に対応した HDF5 ライブラリを入手してください。

表 2.1-1　HDF5 ライブラリ

プラットフォーム	ダウンロードファイル名	備考
All Platform	src/hdf5-1.8.4-patch1.tar.gz	ソースコード
Linux 2.6 i686	hdf5-1.8.4-patch1-linux-shared.tar.gz	バイナリ（共有）
Linux 2.6 x86_64	hdf5-1.8.4-patch1-linux-x86_64-shared.tar.gz	バイナリ（共有）

HDF5 ライブラリは、sz ライブラリを使用しているため、szip-2.1.tar.gz も別途入手が必要です。

表 2.1-2　sz ライブラリ

入手先	備考
ftp://ftp.hdfgroup.org/lib-external/szip/2.1/src/	

2.1.2　HDF5 ライブラリのインストール手順

インストール手順を以下に示します。

(1) ライブラリの入手

コンパイル済みバイナリをインストールする場合は、「Linux 2.6 i686」または「Linux 2.6 x86_64」のバイナリを、ソースコードからコンパイルしインストールする場合は、「All Platform」を入手します。また、「szip-2.1.tar.gz」ファイルを別途入手します。

(2) sz ライブラリのインストール

szip-2.1.tar.gz ファイルを展開し、コンパイルしてインストールします。以下に sz ライブラリを/usr/local/lib にインストールする例を示します。

```
$ tar xzf szip-2.1.tar.gz
$ cd szip-2.1/
$ ./configure –prefix=/usr/local
$ make
スーパーユーザになります。
# make install
```

続いて、HDF5ライブラリをコンパイル済みバイナリからインストールする場合は(3)を、ソースコードからインストールする場合は、(4) を参照して下さい。

(3) HDF5 ライブラリ（バイナリ）のインストール

HDF5ライブラリを解凍し、作業領域に展開します。ここでは、「Linux 2.6 i686」プラットフォームのファイルを例に説明します。

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

上記コマンドを実行するとhdf5-1.8.4-patch1-linux-sharedというディレクトリが作成されます。このディレクトリには、表 2.1-1のファイルとディレクトリが展開されています。

表 2.1-1　HDF ライブラリ

名称	ファイル／ディレクトリの区別	内容
COPYING	ファイル	Copyright Noticeです。
README	ファイル	簡単な使用方法とszライブラリの説明があります。
RELEASE.txt	ファイル	リリースメモです。
bin	ディレクトリ	HDFのツールのディレクトリです。
include	ディレクトリ	インクルードファイルのディレクトリです。
lib	ディレクトリ	ライブラリのディレクトリです。

AMTKのインストーラでHDFのインクルードファイル、ライブラリのディレクトリを指定しますので、HDF関連ファイルは、どのディレクトリにおいても構いません。AMTKのインストールの説明とあわせるため、ここでは、/HDF5/sharedというディレクトリに置くことにします。

```
# mkdir /HDF5
# mkdir /HDF5/shared
# cd hdf5-1.8.4-patch1-linux-shared
# cp –r include/ /HDF5/shared
# cp –r lib/ /HDF5/shared
```

スーパーユーザでディレクトリを作成します。

HDF 関連ファイルをコピーします。

(4) HDF5 ライブラリ（ソースコード）のインストール

hdf5-1.8.4-patch1.tar.gzファイルを解凍し、ファイルを展開します。

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

上記コマンドを実行するとhdf5-1.8.4-patch1というディレクトリが作成されます。
ここでは、/HDF5/sharedというディレクトリをインストール先に指定し、HDF5ライブラリをインストールする例を示します。

```
# cd hdf5-1.8.4-patch1
# ./configure --prefix=/HDF5/shared
# make
# make install
```

インストール先を指定し、HDF5 ライブラリをコンパイル／インストールします。

○SGI環境のインストール

1.2 （1）動作確認環境に示すSGI(IRIX64 / MIPSpro Compilers)環境でインストールする場合、makeコマンドの後に以下のメッセージが出力され、コンパイルがエラーで中断することがあります。

```
92 errors detected in the compilation of "h5tools.c".
*** Error code 1 (bu21)
*** Error code 1 (bu21)
*** Error code 1 (bu21)
```

コンパイルでエラーが発生したことを示すメッセージです。

この場合は、本項で作成したhdf5-1.8.4-patch1ディレクトリに存在するソースコードを改変することで、問題が解消されることがあります。
ソースコードの改変手順を以下に示します。

1） hdf5-1.8.4-patch1/tools/lib/h5tools_error.hをテキストエディタで開きます。
2） 35行目の一文を以下の通り変更し、保存します。
・変更前: #error "We need __func__ or __FUNCTION__ to test function names!
"
・変更後: #define "We need __func__ or __FUNCTION__ to test function names!
"
3） hdf5-1.8.4-patch1ディレクトリにて、以下のコマンドを実行します。

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

インストール先と CFLAGS に-64 オプションを指定し、HDF5 ライブラリをコンパイル／インストールします。

2.2 AMTK のインストール

2.2.1 Linux 環境の AMTK インストール

(1) ファイルの展開

AMTK_AMSR2_Ver1.13.tar.gzファイルを解凍し、ファイルを展開します。

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

上記コマンドを実行するとAMTK_AMSR2というディレクトリが作成されます。このディレクトリには、表 2.2-1のファイルとディレクトリが展開されています。

表 2.2-1 AMTK の内容

名称	ファイル／ディレクトリの区別	内容	備考
Makefile.in autom4te.cash config.gess config.sub configure configure.in install-sh	ファイル	AMTKのインストーラです。	
include/	ディレクトリ	AMTKのインクルードファイルを格納したディレクトリです。	
lib/	ディレクトリ	AMTKのライブラリが作成されるディレクトリです。	
src/	ディレクトリ	AMTKのソースコードを格納したディレクトリです。	
sample/	ディレクトリ	AMTKを利用したサンプルプログラムを格納したディレクトリです。	
share/	ディレクトリ	AMTKの共有設定ファイルを格納したディレクトリです。	うるう秒ファイル、物理量定義ファイル、L3緯度経度ファイルが格納されています。これらのファイルについては、2.3 AMTKの実行環境設定を参照して下さい。

(2) ライブラリのインストール

AMTKのディレクトリに移動した後、AMTKのインストーラ(configureコマンド)を実行

します。インストーラでは、マシン環境に適合したMakefileを生成します。

```
$ ./configure
```

インストーラで以下のHDF関連ファイルを検索しますので、通常のHDFのインストールであれば、指定する必要はありません。

○HDFライブラリファイル
/*/*/libまたは、/*/*/*/lib配下にあるlibhdf5.a

○HDFインクルードファイル
/*/*/includeまたは、/*/*/*/include配下にあるhdf5.h

前述のパス以外にHDF関連ファイルをインストールされた場合は、configureコマンドの以下の引数で指定してください。

○HDFライブラリファイル

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

○HDFインクルードファイル

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

その後、生成されたMakefileを使用して、makeコマンドを実行します。

```
$ make
```

configureコマンドとmakeコマンドを続けて、実行した例を示します。

```
$ ./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

（コンパイル実行）

$ make
cc -c src/amtk_get.c -g -DDBG -DLINUX -ansi -g -DDBG -o src/amtk_get.o  -I./include
-I/HDF5/shared/include -I./src
cc -c src/amtk_set.c -g -DDBG -DLINUX -ansi -g -DDBG -o src/amtk_set.o  -I./include
-I/HDF5/shared/include -I./src
cc -c src/amtk_hdf.c -g -DDBG -DLINUX -ansi -g -DDBG -o src/amtk_hdf.o  -I./include
-I/HDF5/shared/include -I./src
cc -c src/amtk_latlon.c -g -DDBG -DLINUX -ansi -g -DDBG -o src/amtk_latlon.o  -I./include
-I/HDF5/shared/include -I./src
cc -c src/amtk_scantime.c -g -DDBG -DLINUX -ansi -g -DDBG -o src/amtk_scantime.o  -I./include
-I/HDF5/shared/include -I./src
cc -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
```

ライブラリ作成の確認をします。libディレクトリの下にlibAMSR2.aが作成されていれば、AMTKライブラリの作成は、終了です。

```
$ ls –l /home/amtk/AMTK /lib
合計 208
-rw-r--r-- 1 amtk amtk 204672  7 月  25 19:01 libAMSR2.a
```

2.2.2　UNIX 環境の AMTK インストール

(1) ファイルの展開
AMTK_AMSR2_Ver1.12.tar.gzファイルを解凍し、ファイルを展開します。

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

上記コマンドを実行するとAMTK_AMSR2というディレクトリが作成されます。このディレクトリには、表 2.2-1のファイルとディレクトリが展開されています。

表 2.2-3　AMTK の内容

名称	ファイル／ディレクトリの区別	内容	備考
Makefile.in autom4te.cash config.gess config.sub configure configure.in install-sh	ファイル	AMTKのインストーラです。	
include/	ディレクトリ	AMTKのインクルードファイルがあるディレクトリです。	
lib/	ディレクトリ	AMTKのライブラリが作成されるディレクトリです。	
src/	ディレクトリ	AMTKのソースコードがあるディレクトリです。	
sample/	ディレクトリ	AMTKを利用したサンプルプログラムがあるディレクトリです。	
share/	ディレクトリ	AMTKの共有設定ファイルがあるディレクトリです。	うるう秒ファイル、物理量定義ファイル、L3緯度経度ファイルが格納されています。これらのファイルについては、2.3　AMTKの実行環境設定を参照して下さい。

(2) ライブラリのインストール
AMTKのディレクトリに移動した後、マシン環境に適合したMakefileを作成します。
Makefile.inをコピーし、Makefileの雛形を作成します。

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

エディタにてMakefileの以下の項目を変更してください。

○CC

```
CC= cc　(使用するコンパイラのコマンドを指定してください)
```

○CFLAGS

```
CFLAGS= -DSGI -O -s -64                    (SGI の場合)
```

```
CFLAGS= -DSunOS -xO2 -lnsl                 (SunOS の場合)
```

○HDFディレクトリ

```
HDFINC=/ HDF インクルードファイルが存在するディレクトリ
```

makeコマンドを実行します。下記に実行例を示します。

（コンパイル実行）

```
$ 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
```

ライブラリ作成の確認をします。libディレクトリの下にlibAMSR2.aが作成されていれば、AMTKライブラリの作成は、終了です。

```
$ ls –l /home/amtk/AMTK/lib
合計 208
-rw-r--r-- 1 amtk amtk 204672  7 月  25 19:01 libAMSR2.a
```

2.3 AMTK の実行環境設定

AMTKを使用したアプリケーションの実行には、環境変数の設定が必要になります。環境変数の一覧を表 2.3-1に示します。

表 2.3-1 AMTK 環境変数

名称	内容	備考
LD_LIBRARY_PATH	HDF5ライブラリのパスを指定します。	
L3LATLONFILEDIR	L3の緯度経度ファイルが存在するディレクトリを指定します。	share/data/
AMSR2_LEAP_DATA	うるう秒ファイルを指定します。	share/data/leapsec.dat
GEOPHYSICALFILE	物理量定義ファイルを指定します。	share/data/geophysiocal_file

環境変数は、使用するシェルにより指定が異なります。環境変数は、実行時に個々にコマンドで設定できますが、ログインシェルに設定すると毎回設定する必要がなくなります。ここでは、ログインシェルに指定する方法を示します。

(1) csh の場合

ホームディレクトリにある「.cshrc」ファイルに以下を追加してください。

```
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:/HDF5/shared/lib:/usr/local/lib
setenv L3LATLONFILEDIR (L3 の緯度経度ファイルが存在するディレクトリパス)
setenv AMSR2_LEAP_DATA (うるう秒ファイルパス)
setenv GEOPHYSICALFILE (物理量定義ファイルパス)
```

(2) bash の場合

ホームディレクトリにある「.bashrc」ファイルに以下を追加してください。

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/HDF5/shared/lib:/usr/local/lib
export L3LATLONFILEDIR= (L3 の緯度経度ファイルが存在するディレクトリパス)
export AMSR2_LEAP_DATA= (うるう秒ファイルパス)
export GEOPHYSICALFILE= (物理量定義ファイルパス)
```

2.4　うるう秒ファイルについて

AMTKのうるう秒ファイル(share/data/leapsec.dat)は、2010年8月時点の設定となっています。以後、うるう秒が更新される際に、うるう秒ファイルを差し替える必要があります。

2.5　物理量定義ファイルについて

L2、L3の物理量に関係するデータを定義します。本ファイルを編集することで、AMTK本体に手を加えることなく物理量の定義を追加、変更することができます。

物理量定義ファイルは、下表のフォーマットで記述します。

表 2.5-1　物理量定義ファイルフォーマット

行単位フォーマット		Geophysical Name,Scale Factor,Unit,EQR,PS-N,PS-S
1	Geophysical Name	物理量名。メタデータ「GeophysicalName」に対応する値となります。 (アスキーコード36文字以内)
2	Scale Factor	スケールファクタ。データセット「GeophysicalData」に影響します。 (数字文字列、小数点を含み6文字以内)
3	Unit	単位。データセット「GeophysicalData」に影響します。 (アスキーコード10文字以内)
4	EQR	Projectionが"EQR"の時に使用する緯度経度ファイル名の識別記号。物理量によって異なるL3緯度経度ファイルを識別するために使用します。 ("EQR" / "-")
5	PS-N	Projectionが"PS-N"の時に使用する緯度経度ファイル名の物理量名識別記号。物理量によって異なるL3緯度経度ファイルを識別するために使用します。 ("SIC" / "SND" / "-")
6	PS-S	Projectionが"PS-S"の時に使用する緯度経度ファイル名の物理量名識別記号。物理量によって異なるL3緯度経度ファイルを識別するために使用します。 ("SIC" / "SND" / "-")

- 1行に１物理量を定義します。
- 各データは、半角カンマ(',')区切りとします。
- 設定の必要がないデータは、半角ハイフン('-')を設定します。
- 先頭の文字が半角ハッシュマーク('#')の行はコメント扱いとし、読込対象から除外します。
- 輝度温度(Brightness Temperature)も本ファイルに定義します。ScaleFactor、Unitのデータは、輝度温度固有のデータを適用するため設定不要です。

【物理量定義ファイルの記述例】

```
#Geophysical Name,Scale Factor,Unit,EQR,PS-N,PS-S （コメント文）
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
```

AMTKが提供するデフォルトの物理量定義ファイル(share/data/geophysiocal_file)は、L2、L3のHDFアクセスに必要なメタデータ「GeophysicalName」に設定すべき値が記述されて

います。

メタデータ「GeophysicalName」の値に対応した物理量定義ファイルの内容を、以下の表に示します。

表 2.5-2　物理量定義ファイルの内容

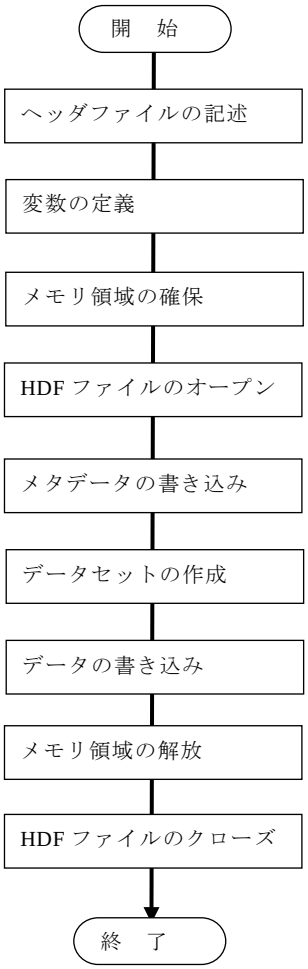
メタデータ GeophysicalName に設定すべき値	内容	Unit	scale factor	EQR	PS-N	PS-S	L2用	L3用
Total Precipitable Water	積算水蒸気量	kg/m2	0.01	EQR	-	-	○	○
Cloud Liquid Water	積算雲水量	kg/m2	0.001	EQR	-	-	○	○
Precipitation	降水量	mm/h	0.01	EQR	-	-	○	○
Sea Surface Temperature	海面温度	C	0.01	EQR	-	-	○	○
Sea Surface Wind speed	海上風	m/s	0.01	EQR	-	-	○	○
Sea Ice Concentration	海氷密接度	%	0.1	-	SIC	SIC	○	○
Snow Depth	積雪	cm	0.1	EQR	SND	-	○	○
Soil Moisture Content	土壌水分	%	0.1	EQR	-	-	○	○
Brightness Temperature (6GHz)	輝度温度 6GHz	- ※不変	- ※不変	EQR	SIC	SIC	-	○
Brightness Temperature (7GHz)	輝度温度 7GHz	- ※不変	- ※不変	EQR	SIC	SIC	-	○
Brightness Temperature (10GHz)	輝度温度 10GHz	- ※不変	- ※不変	EQR	SIC	SIC	-	○
Brightness Temperature (18GHz)	輝度温度 18GHz	- ※不変	- ※不変	EQR	SIC	SIC	-	○
Brightness Temperature (23GHz)	輝度温度 23GHz	- ※不変	- ※不変	EQR	SIC	SIC	-	○
Brightness Temperature (36GHz)	輝度温度 36GHz	- ※不変	- ※不変	EQR	SIC	SIC	-	○
Brightness Temperature (89GHz)	輝度温度 89GHz	- ※不変	- ※不変	EQR	SIC	SIC	-	○

3　AMTK を利用したプログラミング

AMTKを使って、AMSR2データをCやFortranで読み書きする流れを以下に示します。ここでは、プログラミングの流れを示し、実際のソースは、C言語、Fortranのサンプルプログラムを参考にしてください。

3.1　プログラミングの流れ

(1) HDF ファイルの新規作成



AMTK のヘッダファイルをインクルードします。

使用する変数を定義します。

書き込むデータのメモリ領域を確保します。

HDF ファイルを書き込みモードでオープンします。HDF ファイルが存在していない場合は、新規に作成されます。

メタデータを HDF ファイルに書き込みます。

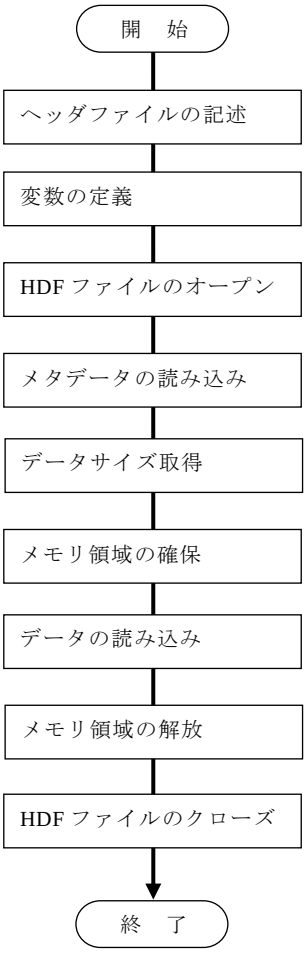
データセットを HDF ファイルに作成します。

データセットにデータをに書き込みます。

メモリ領域を解放します。

HDF ファイルをクローズします。

(2) HDF ファイルからデータの読み出し



AMTK のヘッダファイルをインクルードします。

使用する変数を定義します。

HDF ファイルを読み込みモードでオープンします。

メタデータを HDF ファイルから読み込みます。

読み込むデータセットのサイズを取得します。

読み込むデータのメモリ領域を確保します。

データセットからデータを読み込みます。

メモリ領域を解放します。

HDF ファイルをクローズします。

3.2　C プログラミング

3.2.1　C サンプルプログラム

C 言語のサンプルプログラムを表3.2-1に示します。

表 3.2-2 C サンプルプログラム

ファイル名	サンプルプログラムの説明	備考
sample1.c	メタデータと Navigation Data データセットを作成し、読み込みます。	プログラムは、3.2.2 章で説明しています。 実行形式名は、sample1 です。
sample2_make_L1Bproduct.c	L1B のメタデータ及びデータセットを作成します。	実行形式名は、sample2 です。
sample3_make_L2Lproduct.c	L2 低解像度のメタデータ及びデータセットを作成します。	実行形式名は、sample3 です。
sample4_make_L2Hproduct.c	L2 高解像度のメタデータ及びデータセットを作成します。	実行形式名は、sample4 です。
sample5_make_L3Dproduct.c	L3 日単位のメタデータ及びデータセットを作成します。	実行形式名は、sample5 です。
sample6_make_L3Mproduct.c	L3 月単位のメタデータ及びデータセットを作成します。	実行形式名は、sample6 です。
sample7_ read_L1Bproduct.c	L1B のメタデータ及びデータセットを作成します。	実行形式名は、sample7 です。
sample8_read_L2Lproduct.c	L2 低解像度のメタデータ及びデータセットを画面に出力します。	実行形式名は、sample8 です。
sample9_read_L2Hproduct.c	L2 高解像度のメタデータ及びデータセットを画面に出力します。	実行形式名は、sample9 です。
sample10_read_L3Dproduct.c	L3 日単位のメタデータ及びデータセットを画面に出力します。	実行形式名は、sample10 です。
sample11_read_L3Mproduct.c	L3 月単位のメタデータ及びデータセットを画面に出力します。	実行形式名は、sample11 です。
sample12_make_L1Rproduct.c	L1R のメタデータ及びデータセットを作成します。	実行形式名は、sample12 です。
sample13_read_L1Rproduct.c	L1R のメタデータ及びデータセットを画面に出力します。	実行形式名は、sample13 です。
sample_common.c	サンプルプログラム用の共通関数です。	-
sample_common.h	サンプルプログラム用の共通ヘッダです。	-

詳細は、サンプルプログラムがあるディレクトリのREADME.txtを参照してください。

3.2.2 サンプルプログラムの説明

ここでは、sample1.c より、HDF に関連する処理を説明します。

(1) HDF ファイルの新規作成

(a) ヘッダファイルの記述

`#include "AMTK.h"`	AMTK のヘッダファイルをインクルードします。

(b) 変数の定義

`hid_t file_id = 0;`	ファイル ID を格納する変数を定義します。

`typedef struct`
`{`
`...`
`  /* Navigation Data[OUT]. */`
`  float *p_navi_out;`
 	
`} DATASET_MODEL;`
`...`
`DATASET_MODEL dataset = {0};`
	出力するデータセット Navigation Data の値を格納する変数を宣言します。
	ここでは動的にメモリ領域を確保するため、float 型のポインタとします。

`int dimsize[3] = {0};`	データセットのサイズを指定する変数を定義します。

(c) メモリ領域の確保

`dataset.p_navi_out = (float *) malloc(sizeof(float) * scan_size * navi_num);`	データセット Navigation Data のメモリ領域を確保します。

(d) HDF ファイルのオープン

`file_id = AMTK_openH5_Write(FN_SAMPLE_PRODUCT_L1B, AM2_CREATE_MODE);`	HDF ファイルを新規作成モードでオープンします。

(e) メタデータの書き込み

`ret = AMTK_setMetaDataName(file_id, META_PRODUCTNAME_NAME, META_PRODUCTNAME_VAL);`	メタデータ ProductName に文字列 "AMSR2-L1B"を書き込みます。
`ret = AMTK_setMetaDataName(file_id, META_OVERLAP_NAME, META_OVERLAP_VALUE);`	メタデータ OverlapScans に文字列 "20"を書き込みます。

(f) データセットの作成

`dimsize[0] = scan_size;`
`dimsize[1] = navi_num;`
`dimsize[2] = 0;`
	データセットのサイズを設定します。
	ここでは、Navigation Data を 2040 x 6 (2 次元データ)のサイズとします。

`ret = AMTK_setDimSize(file_id, AM2_NAVI, dimsize);`	上記のサイズでデータセット Navigation Data を作成します。

(g) データの書き込み

ret = AMTK_set_SwathFloat(file_id, navi, scan_start, scan_end, AM2_NAVI);	データセット Navigation Data に float 型の配列データを書き込みます。 スキャン開始位置、終了位置はオーバラップ数を含めた全スキャンを指定します。詳細は 5.3　スキャン番号を参照してください。

(h) メモリ領域の解放

free(p_dataset->p_navi_out);	データタセット Navigation Data のメモリ領域を解放します。

(i) HDF ファイルのクローズ

ret = AMTK_closeH5_Write(file_id);	HDF ファイルをクローズします。

(2) HDF ファイルからデータの読み込み

(a) ヘッダファイルの記述

#include "AMTK.h"	AMTK のヘッダファイルをインクルードします。

(b) 変数の定義

hid_t file_id = 0;	ファイル ID を格納する変数を定義します。
typedef struct { /* Navigation Data[IN]. */ float *p_navi_in; ... } DATASET_MODEL; ... DATASET_MODEL dataset = {0};	入力するデータセット Navigation Data の値を格納する変数を定義します。 ここでは動的にメモリ領域を確保するため、float 型のポインタとします。
int dimsize[3] = {0};	データセットのサイズを指定する変数を定義します。

(c) HDF ファイルのオープン

file_id = AMTK_openH5(FN_SAMPLE_PRODUCT_L1B);	HDF ファイルをオープンします。

(d) メタデータの読み込み

ret = AMTK_getMetaDataName(file_id, META_PRODUCTNAME_NAME, &p_value);	メタデータ ProductName の値を p_value に読み込みます。

(e) データサイズ取得

ret = AMTK_getDimSize(file_id, AM2_NAVI, dimsize);	データセット Navigation Data のサイズを取得します。 ここでは、出力時に設定された 2040 x 6 のサイズが dimsize に設定されます。

(f) メモリ領域の確保

dataset.p_navi_in = (float *) malloc(sizeof(float) * get_memory_size(dimsize));	Navigation Data のメモリ領域を確保します。型のサイズ(float) * dimsize[0] * dimsize[1] * dimsize[2](サイズ 0 の次元は乗算対象外)を必要とします。 get_memory_size()は、前述の dimsize の計算を行う内部関数です。(sample_common.c に実装)

(g) データの読み込み

ret = AMTK_get_SwathFloat(file_id, &dataset.p_navi_in, scan_start, scan_end, AM2_NAVI);	データセット Navigation Data から float 型の配列データを読み込みます。 スキャン開始位置、終了位置はオーバラップ数を含めた全スキャンを指定します。詳細は 5.3　スキャン番号を参照してください。

(h) メモリ領域の解放

free(p_dataset->p_navi_in);	データタセット Navigation Data のメモリ領域を解放します。

(i) HDF ファイルのクローズ

ret = AMTK_closeH5(file_id);	HDF ファイルをクローズします。

3.2.3　サンプルプログラムの内容

3.2.2 で説明した sample1.c を以下に示します。

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

/** Include AMTK.h */
#ifdef __cplusplus
extern "C" {
#endif
#include "AMTK.h"
#ifdef __cplusplus
}
#endif

#include "sample_common.h"

/*---------------------------------------------------------------------------*
 * Common definition                                                         *
 *---------------------------------------------------------------------------*/
#define FN_SAMPLE_PRODUCT_L1B   ("../data/sample1_IOToolKit.h5")

#define META_PRODUCTNAME_NAME   ("ProductName")
#define META_PRODUCTNAME_VAL    ("AMSR2-L1B")
#define META_OVERLAP_NAME       ("OverlapScans")
#define META_OVERLAP_VALUE      ("20")

#define SCENE_SCAN_NUM          (2000)
#define OVERLAP_SCAN_NUM        (20)


/** Structure of dataset model. */
typedef struct
{
      /* Navigation Data[IN]. */
      float *p_navi_in;

      /* Navigation Data[OUT]. */
      float *p_navi_out;

} DATASET_MODEL;

/** File R/W flag. */
enum FLAG_FILERW
{
      FILE_READ,
      FILE_WRITE
};


/** Termination.
 * @param file_id   [IN]  Product file id
 * @param is_write  [IN]  Write mode flag
 * @param p_dataset [OUT] Dataset
 */
static void terminate(hid_t file_id, int is_write, DATASET_MODEL *p_dataset);


/** AMSR2 I/O toolkit sample program.
 * sample1: proudct I/O.
 *
 * @param  argc [IN] Argument count
 * @param  argv [IN] Argument value
```

```
 * @return EXIT_SUCCESS
 * @return EXIT_FAILURE
 */
int main(int argc, char *argv[])
{
	hid_t file_id = 0;
	int dimsize[3] = {0};
	int i = 0;
	int j = 0;
	int ret = 0;
	char productname[12 + 1] = "";
	char *p_value = NULL;

	/* Number of scans */
	const int scan_size = SCENE_SCAN_NUM + OVERLAP_SCAN_NUM * 2;
	const int scan_start = 1 - OVERLAP_SCAN_NUM;
	const int scan_end = SCENE_SCAN_NUM + OVERLAP_SCAN_NUM;
	const int navi_num = 6;

	/* Dataset */
	DATASET_MODEL dataset = {0};

	/*-------------------------------------------------------------------------*
	 * Output product                                                          *
	 *-------------------------------------------------------------------------*/
	dataset.p_navi_out = (float *) malloc(sizeof(float) * scan_size * navi_num);
	if (NULL == dataset.p_navi_out)
	{
		E_MSG("malloc() error.¥n");
		exit(EXIT_FAILURE);
	}

	/* file open */
	file_id = AMTK_openH5_Write(FN_SAMPLE_PRODUCT_L1B, AM2_CREATE_MODE);
	if (0 > file_id)
	{
		E_MSG("AMTK_openH5_Write() error.[%d]¥n", file_id);
		terminate(file_id, FILE_WRITE, &dataset);
		exit(EXIT_FAILURE);
	}

	/* set metadata */
	ret = AMTK_setMetaDataName(file_id, META_PRODUCTNAME_NAME,
		META_PRODUCTNAME_VAL);
	if (0 > ret)
	{
		E_MSG("AMTK_setMetaDataName() error.[%d]¥n", ret);
		terminate(file_id, FILE_WRITE, &dataset);
		exit(EXIT_FAILURE);
	}

	ret = AMTK_setMetaDataName(file_id, META_OVERLAP_NAME, META_OVERLAP_VALUE);
	if (0 > ret)
	{
		E_MSG("AMTK_setMetaDataName() error.[%d]¥n", ret);
		terminate(file_id, FILE_WRITE, &dataset);
		exit(EXIT_FAILURE);
	}

	/* set dataset */
	dimsize[0] = scan_size;
	dimsize[1] = navi_num;
	dimsize[2] = 0;
	ret = AMTK_setDimSize(file_id, AM2_NAVI, dimsize);
	if (0 > ret)
	{
		E_MSG("AMTK_setMetaDataName() error.[%d]¥n", ret);
```

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

        /* set Navigation Data */
        for (i = 0; i < scan_size; i++)
        {
                for (j = 0; j < navi_num; j++)
                {

                        *(dataset.p_navi_out + (i * navi_num) + j)
                                = (i - OVERLAP_SCAN_NUM) * 0.1;
                }
        }

        ret = AMTK_set_SwathFloat(file_id, dataset.p_navi_out, scan_start,
                scan_end, AM2_NAVI);
        if (0 > ret)
        {
                E_MSG("AMTK_set_SwathFloat() error.[%d]¥n", ret);
                terminate(file_id, FILE_WRITE, &dataset);
                exit(EXIT_FAILURE);
        }

        /* file close */
        ret = AMTK_closeH5_Write(file_id);
        if (0 > ret)
        {
                E_MSG("AMTK_closeH5_Write() error.[%d]¥n", ret);
                terminate(file_id, FILE_WRITE, &dataset);
                exit(EXIT_FAILURE);
        }

        file_id = -1;

        /*-------------------------------------------------------------------------*
         * Input product                                                           *
         *-------------------------------------------------------------------------*/
        /* file open */
        file_id = AMTK_openH5(FN_SAMPLE_PRODUCT_L1B);
        if (0 > file_id)
        {
                E_MSG("AMTK_openH5() error.[%d]¥n", file_id);
                terminate(file_id, FILE_READ, &dataset);
                exit(EXIT_FAILURE);
        }

        /* get dimension size */
        ret = AMTK_getDimSize(file_id, AM2_NAVI, dimsize);
        if (0 > ret)
        {
                E_MSG("AMTK_getDimSize() error.[%d]¥n", ret);
                terminate(file_id, FILE_READ, &dataset);
                exit(EXIT_FAILURE);
        }

        /* memory allocation */
        dataset.p_navi_in = (float *) malloc(sizeof(float) *
                get_memory_size(dimsize));
        if (NULL == dataset.p_navi_in)
        {
                E_MSG("malloc() error.¥n");
                terminate(file_id, FILE_READ, &dataset);
                exit(EXIT_FAILURE);
        }
```

```
        /* get metadata */
        p_value = productname;
        ret = AMTK_getMetaDataName(file_id, META_PRODUCTNAME_NAME, &p_value);
        if (0 > ret)
        {
                E_MSG("AMTK_getMetaDataName() error.[%d]¥n", ret);
                terminate(file_id, FILE_READ, &dataset);
                exit(EXIT_FAILURE);
        }

        /* get dataset */
        ret = AMTK_get_SwathFloat(file_id, &dataset.p_navi_in, scan_start, scan_end,
                AM2_NAVI);
        if (0 > ret)
        {
                E_MSG("AMTK_get_SwathFloat() error.[%d]¥n", ret);
                terminate(file_id, FILE_READ, &dataset);
                exit(EXIT_FAILURE);
        }

        printf("ProductName = %s Navigation Data[0] = %f ¥n", p_value,
                *(dataset.p_navi_in));

        /* termination */
        terminate(file_id, FILE_READ, &dataset);

        exit(EXIT_SUCCESS);
}



/** Termination.
 * @param file_id   [IN]  Product file id
 * @param is_write  [IN]  Write mode flag
 * @param p_dataset [OUT] Dataset
 */
static void terminate(hid_t file_id, int is_write, DATASET_MODEL *p_dataset)
{
        int ret = 0;

        if (0 <= file_id)
        {
                /* file close */
                if (is_write)
                {
                        /* Read/Write mode */
                        ret = AMTK_closeH5_Write(file_id);
                        if (0 > ret)
                        {
                                E_MSG("AMTK_closeH5_Write() error.[%d]¥n", ret);
                        }
                }
                else
                {
                        /* Read mode */
                        ret = AMTK_closeH5(file_id);
                        if (0 > ret)
                        {
                                E_MSG("AMTK_closeH5() error.[%d]¥n", ret);
                        }
                }
        }

        free(p_dataset->p_navi_in);
        p_dataset->p_navi_in = NULL;

        free(p_dataset->p_navi_out);
```

```
        p_dataset->p_navi_out = NULL;

        return;
}
```

3.2.4　コンパイル及び実行

最初にmakeコマンド実行のためのMakefileを作成します。

```
$ ./configure
```

AMTKインストール時と同じく、HDF関連ファイルを自動で検索しMakefileを生成します。また、configureコマンドのオプション「--with-hdf-lib」「--with-hdf-include」を使用することで、AMTKインストール時と同様にHDF関連ファイルのパスを指定することが可能です。

コンパイラを変更したい場合や、HDF関連ファイルのパスを編集したい場合は、configureコマンドで生成されたMakefileの定義をエディタで修正してください。

- コンパイルコマンド

```
CC=使用コンパイラのコンパイルコマンド
CFLAGS=使用コンパイラのオプション(SGI の場合は-64 必須)
```

- HDFライブラリ

```
HDFLIB=/ HDF ライブラリが存在するディレクトリ
```

- HDFディレクトリ

```
HDFINC=/ HDF インクルードファイルが存在するディレクトリ
```

実行形式作成のmakeコマンドを実行します。

```
$ make
```

makeコマンドを実行すると現在のディレクトリに全アンプルプログラムの実行形式が作成されます。各サンプルプログラムを実行します。以下は、sample1の実行の例を示します。

```
$./sample1
ProductName = AMSR2-L1B Navigation Data[0] = -2.000000
```

sample1を実行すると上記のメッセージが標準出力され、sample/dataディレクトリにsample1_IOToolKit.h5というHDFファイルが作成されます。

他のサンプルもコンパイル及び実行は同様です。

3.3 Fortran プログラミング

AMTKが提供するFortranの機能は、Fortran95用とFortran77用の2種類が存在します。
サンプルプログラムは、sample/Fortranディレクトリに Fortran95用、sample/Fortran77 ディレクトリにFortran77用のソースコードを格納しています。
ここでは、Fortran95用のプログラミングについて説明します。ファイル構成と実行方法は、Fortran77においても同じです。

3.3.1 Fortran サンプルプログラム

Fortran のサンプルプログラムを表に示します。

表 3.3-1 Fortran サンプルプログラム

ファイル名	サンプルプログラムの説明	備考
sample1.f	メタデータを２つとナビゲーションのデータセットを作成し、再度読み込みます。	プログラムは、3.3.2 章で説明しています。 実行形式名は、sample1 です。
sample2_make_L1Bproduct.f	L1B のメタデータ及びデータセットを作成します。	実行形式名は、sample2 です。
sample3_make_L2Lproduct.f	L2 低解像度のメタデータ及びデータセットを作成します。	実行形式名は、sample3 です。
sample4_make_L2Hproduct.f	L2 高解像度のメタデータ及びデータセットを作成します。	実行形式名は、sample4 です。
sample5_make_L3Dproduct.f	L3 日単位のメタデータ及びデータセットを作成します。	実行形式名は、sample5 です。
sample6_make_L3Mproduct.f	L3 月単位のメタデータ及びデータセットを作成します。	実行形式名は、sample6 です。
sample7_ read_L1Bproduct.f	L1B のメタデータ及びデータセットを作成します。	実行形式名は、sample7 です。
sample8_read_L2Lproduct.f	L2 低解像度のメタデータ及びデータセットを画面に出力します。	実行形式名は、sample8 です。
sample9_read_L2Hproduct.f	L2 高解像度のメタデータ及びデータセットを画面に出力します。	実行形式名は、sample9 です。
sample10_read_L3Dproduct.f	L3 日単位のメタデータ及びデータセットを画面に出力します。	実行形式名は、sample10 です。
sample11_read_L3Mproduct.f	L3 月単位のメタデータ及びデータセットを画面に出力します。	実行形式名は、sample11 です。
sample12_make_L1Rproduct.f	L1R のメタデータ及びデータセットを作成します。	実行形式名は、sample12 です。
sample13_read_L1Rproduct.f	L1R のメタデータ及びデータセットを画面に出力します。	実行形式名は、sample13 です。

詳細は、サンプルプログラムがあるディレクトリのREADME.txtを参照してください。

3.3.2 サンプルプログラムの説明

ここでは、sample1.f より、HDF に関連する処理を説明します。

(1) HDF ファイルの新規作成

(a) ヘッダファイルの記述

include 'AMTK_f.h'	AMTK ライブラリのヘッダファイル

(b) 変数の定義

Integer file_id	AMTK ライブラリのヘッダファイル
real navi(6, 2040)	出力するデータセット Navigation Data の値を格納する変数を宣言します。 Fortran は、作成するデータセットのサイズと配列の次元の順序が逆になります。サイズと配列と関係は、5.2.1 (1) AMTK_getDimSize の説明を参照して下さい。ここでは、Navigation Data を 2040 x 6 (2 次元データ)サイズとします。
Integer size(3)	データセットのサイズを指定する変数を定義します。

(c) HDF ファイルのオープン

file_id = AMTK_openH5_Write(fname, AM2_CREATE_MODE)	HDF ファイルを新規作成モードでオープンします。

(d) メタデータの書き込み

status = AMTK_setMetaDataName(file_id, name_productname, val_productname)	メタデータ ProductName に文字列 "AMSR2-L1B"を書き込みます。
status = AMTK_setMetaDataName(file_id, name_overlapscans, val_overlapscans)	メタデータ OverlapScans に文字列 "20"を書き込みます。

(e) データセットの作成

size(1) = scan_size size(2) = 6 size(3) = 0	データセットのサイズを指定します。 ここでは、Navigation Data を 2040 x 6 (2 次元データ)のサイズとします。
status = AMTK_setDimSize(file_id,AM2_NAVI, size)	上記のサイズでデータセット Navigation Data を作成します。。

(f) データの書き込み

status = AMTK_set_SwathFloat(file_id, navi, scan_start, scan_end, AM2_NAVI)	データセット Navigation Data に Real 型の配列データを書き込みます。 スキャン開始位置、終了位置はオーバラップ数を含めた全スキャンを指定します。詳細は 5.3 スキャン番号を参照してください。

(g) HDF ファイルのクローズ

status = AMTK_closeH5_Write(file_id)	HDF ファイルをクローズします。

(2) HDF ファイルからデータの読み込み

(a) HDF ファイルのオープン

file_id = AMTK_openH5(fname);	HDF ファイルをオープンします。

(b) メタデータの読み込み

status = AMTK_getMetaDataName(file_id, name_productname, meta_value)	メタデータ ProductName の値を meta_value に読み込みます。

(c) データの読み込み

status = AMTK_get_SwathFloat(file_id, navi, scan_start, scan_end, AM2_NAVI)	データセット Navigation Data から Real 型の配列データを読み込みます。 スキャン開始位置、終了位置はオーバラップ数を含めた全スキャンを指定します。詳細は 5.3　スキャン番号を参照してください。

(d) HDF ファイルのクローズ

status = AMTK_closeH5(file_id)	HDF ファイルをクローズします。

3.3.3 サンプルプログラムの内容

3.3.2 で説明した sample1.f の内容を以下に示します。

```
      Program main

      Implicit NONE

      include 'AMTK_f.h'

      character*40 fname
      data fname/'../data/sample1_IOToolKit_f.h5'/
      Integer file_id
      Integer status

      character*11 name_productname
      data name_productname/'ProductName'/
      character*9 val_productname
      data val_productname/'AMSR2-L1B'/

      character*12 name_overlapscans
      data name_overlapscans/'OverlapScans'/
      character*2 val_overlapscans
      data val_overlapscans/'20'/

      character*20 meta_value
      data meta_value/''/
      Integer size(3)
      Integer i
      Integer j

      real navi(6, 2040)

C     Scene Scan number
      integer, PARAMETER :: SCENE_SCAN_NUM=2000
C     Overlap Scans
      integer, PARAMETER :: OVERLAP_SCAN_NUM=20
      Integer scan_start
      Integer scan_end
      Integer scan_size

      scan_size = SCENE_SCAN_NUM + OVERLAP_SCAN_NUM * 2
      scan_start = 1 - OVERLAP_SCAN_NUM
      scan_end = SCENE_SCAN_NUM + OVERLAP_SCAN_NUM

C-----------------------------------------------------------------------C
C    *Output product                                                    C
C-----------------------------------------------------------------------C
C     file open
      file_id = AMTK_openH5_Write(fname , AM2_CREATE_MODE)
      if (file_id.lt.0) then
          write(*,*) 'AMTK_openH5_Write ERROR' ,file_id
          stop
      endif

C     set metadata
      status = AMTK_setMetaDataName(file_id, name_productname,
     *    val_productname)
      if (status.lt.0) then
          write(*,*) 'AMTK_setMetaDataName ERROR' ,status
          stop
      endif

      status = AMTK_setMetaDataName(file_id, name_overlapscans,
```

```
     *    val_overlapscans)
      if (status.lt.0) then
          write(*,*) 'AMTK_setMetaDataName ERROR',status
          stop
      endif

C     set dataset

C     Dimension size <-> Array size
C     Elements of the array is reversed.
C      Example)
C       AMTK_getDimSize(*, *, dim_size)
C       dim_size(1) : Scan
C       dim_size(2) : Pixel
C       -> array(Pixel, Scan)

      size(1) = scan_size
      size(2) = 6
      size(3) = 0

      status = AMTK_setDimSize(file_id, AM2_NAVI, size)
      if (status.lt.0) then
          write(*,*) 'AMTK_setDimSize ERROR',status
          stop
      endif

      Do i = 1, 6
          Do j = 1, scan_size
              navi(i, j) = (j - OVERLAP_SCAN_NUM - 1) * 0.1
          End do
      End do

      status = AMTK_set_SwathFloat(file_id, navi, scan_start,
     *    scan_end, AM2_NAVI)
      if (status.lt.0) then
          write(*,*) 'AMTK_set_SwathFloat ERROR',status
          stop
      endif

C     file close
      status = AMTK_closeH5_Write(file_id)
      if (status.lt.0) then
          write(*,*) 'AMTK_closeH5_Write ERROR',status
          stop
      endif

C-----------------------------------------------------------------------C
C    *Input product                                                     C
C-----------------------------------------------------------------------C
C     file open
      file_id = AMTK_openH5(fname)
      if (file_id.lt.0) then
          write(*,*) 'AMTK_openH5 ERROR',file_id
          stop
      endif

C     get metadata
      status = AMTK_getMetaDataName(file_id, name_productname,
     *    meta_value)
      if (status.lt.0) then
          write(*,*) 'AMTK_getMetaDataName ERROR',status
          stop
      endif

C     get dataset
      status = AMTK_get_SwathFloat(file_id, navi, scan_start,
     *    scan_end, AM2_NAVI)
```

```
        if (status.lt.0) then
            write(*,*) 'AMTK_get_SwathFloat ERROR' ,status
            stop
        endif

C       file close
        status = AMTK_closeH5(file_id)
        if (status.lt.0) then
            write(*,*) 'AMTK_closeH5 ERROR' ,status
            stop
        endif

        Write(*,*) 'ProductName = ', meta_value,
       *    ' Navigation Data(1, 1) = ', navi(1, 1)

        Stop
        End
```

3.3.4 コンパイル及び実行

最初にmakeコマンド実行のためのMakefileを作成します。

```
$ ./configure
```

AMTKインストール時と同じく、HDF関連ファイルを自動で検索しMakefileを生成します。また、configureコマンドのオプション「--with-hdf-lib」「--with-hdf-include」を使用することで、AMTKインストール時と同様にHDF関連ファイルのパスを指定することが可能です。

コンパイラを変更したい場合や、HDF関連ファイルのパスを編集したい場合は、configureコマンドで生成されたMakefileの定義をエディタで修正してください。

- コンパイルコマンド

  FC=使用コンパイラのコンパイルコマンド
  CFLAGS=使用コンパイラのオプション(SGI の場合は-64 必須)

- HDFライブラリ

  HDFLIB=/ HDF ライブラリが存在するディレクトリ

- HDFディレクトリ

  HDFINC=/ HDF インクルードファイルが存在するディレクトリ

実行形式作成のmakeコマンドを実行します。

```
$ make
```

makeコマンドを実行すると現在のディレクトリに全アンプルプログラムの実行形式が作

成されます。各サンプルプログラムを実行します。以下は、sample1の実行の例を示します。

```
$./sample1
ProductName = AMSR2-L1B Navigation Data(1,1) = -2.0
```

sample1を実行すると上記のメッセージが標準出力され、sample/dataディレクトリにsample1_IOToolKit_f.h5というHDFファイルが作成されます。

他のサンプルもコンパイル及び実行は同様です。

4　機能構成

AMTK の機能構成図を図4-1に示します。

提供機能は、入力機能、出力機能で分けられます。

1）　入力機能

(ア) レベル１プロダクト（L1A、L1B、L1R）入力機能

(イ) レベル２プロダクト（L2）入力機能

(ウ) レベル３プロダクト（L3）入力機能

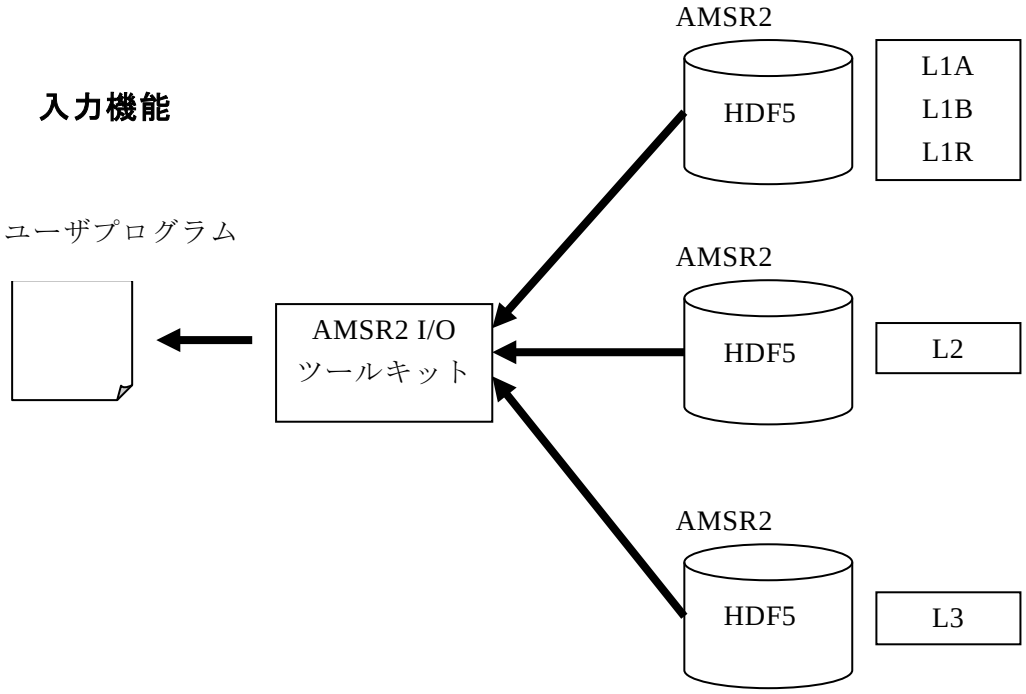


図 4-2 AMTK　プロダクト入力機能

２）　出力機能

(ア) レベル１プロダクト（L1A、L1B、L1R）出力機能

(イ) レベル２プロダクト（L2）出力機能

(ウ) レベル３プロダクト（L3）出力機能

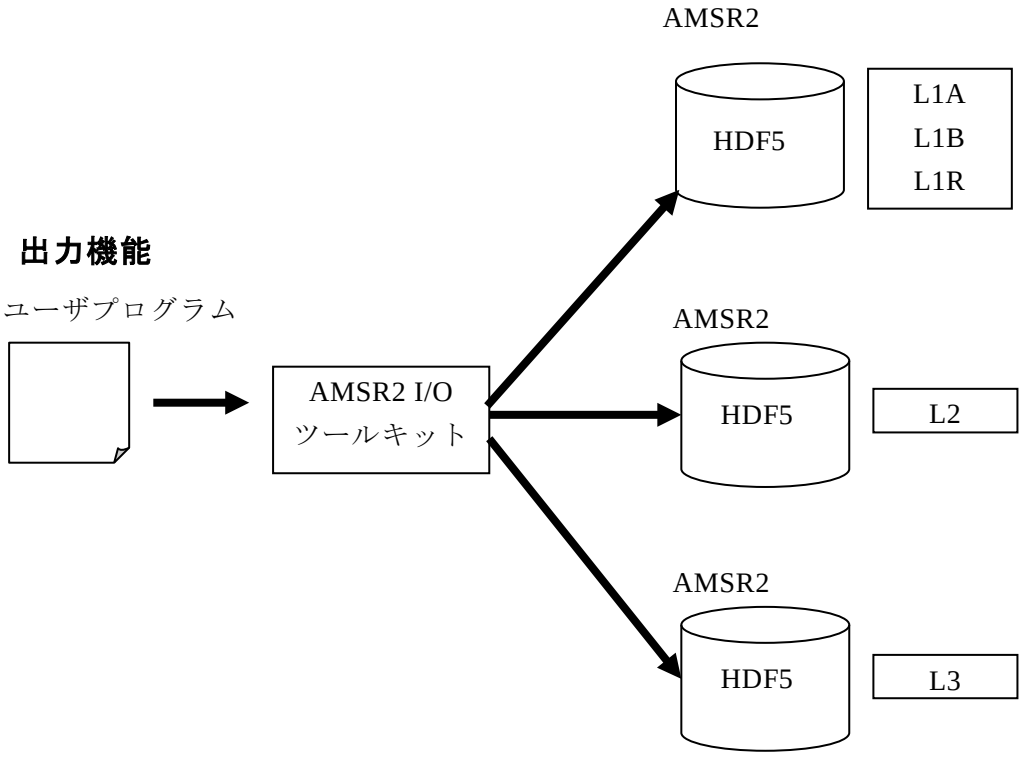


図 4-2　AMTK　プロダクト出力機能

5　AMTK の関数

5.1　C 言語

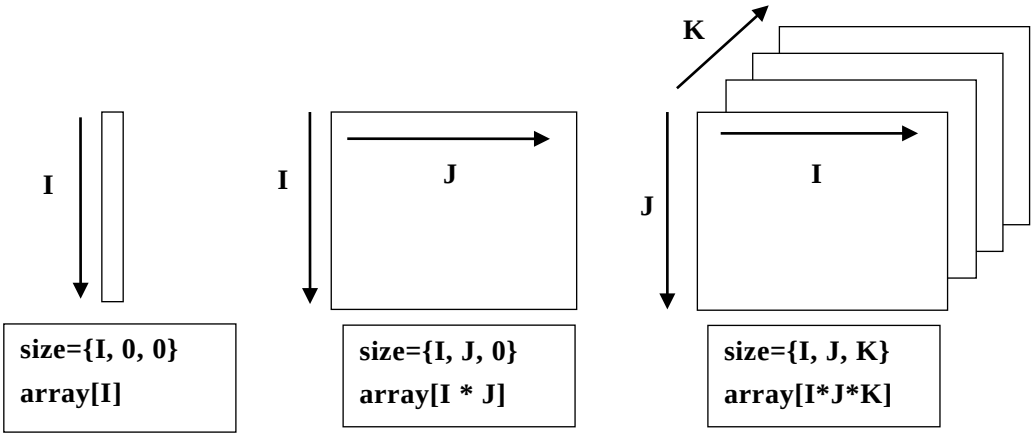
5.1.1　共通関数

(1) L1A、L1B、L1R、L2、L3 入力機能

プロダクトファイルオープン(読込み専用)				
HDFプロダクトファイルを読込み専用でオープンします。				
hid_t AMTK_openH5(char *file_name)				
名前	型	入出力区別	サイズ	説明
戻り値				
HDF_file_id	hid_t	output	1	正常時:HDF access file idが返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
file_name	char *	input	1	AMSR2 HDFファイル名

プロダクトファイルクローズ				
HDFプロダクトファイルをクローズします。				
int AMTK_closeH5(hid_t HDF_file_id)				
名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:0 異常時:負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id

データディメンジョンサイズ取得				
プロダクトからデータディメンジョンサイズを取得します。 **ー　複数のデータセットを対象とするアクセルラベルが指定された場合は、エラーを返します。**				
int AMTK_getDimSize(hid_t HDF_file_id, int access_lbl , int *size)				
名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	**正常時：配列次元数が返ります。** **異常時：取得できない場合は、負の値が返ります。**
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
access_lbl	int	input	1	**「6.1.1　HDFアクセスラベル」を参照してください。**
size	int *	output	1	**ディメンジョンサイズ（サイズと配列の関係は下図を参照してください。）**



一次元データ

スキャン数: **I**

二次元データ

スキャン数: **I**

三次元データ

スキャン数: **J**

「6.1.2　物理量に関するデータセットの説明」も併せて参照してください。

メタデータ取得(メタデータ名)

プロダクトのメタデータ名でメタデータを取得します。

ー メタデータ値を設定するメモリ領域がNULLの場合は、エラーを返します。

int AMTK_getMetaDataName(hid_t HDF_file_id, char *name, char **value)

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:メタデータ文字数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
name	char *	input	1	メタデータ名
value	char **	output	1	メタデータ値

メタデータ取得(インデックス値)

プロダクトのインデックス値でメタデータを取得します。

ー メタデータ値を設定するメモリ領域がNULLの場合は、エラーを返します。

int AMTK_getMetaData(hid_t HDF_file_id, int index, char **value)

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:メタデータ文字数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
index	int	input	1	メタデータインデックス値
value	char **	output	1	メタデータ値

スキャン時刻取得

スキャン時刻を取得します。

- **スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。**
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを取得します。
- **データを格納するメモリエリアは、関数を使用するユーザが確保します。**
  データサイズは、「sizeof(AM2_COMMON_SCANTIME)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) 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)

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
from_scan	int	input	1	取得開始スキャン番号
to_scan	int	input	1	取得終了スキャン番号
scan_time	AM2_COMMON_SCANTIME **	output	1	「6.2 L1、L2、L3共通データ」を参照してください。

緯度経度データ取得

指定されたスキャン番号の範囲の緯度経度データを取得します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを取得します。
- データを格納するメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「sizeof(AM2_COMMON_LATLON)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) 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)

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
latitudelongitude	AM2_COMMON_LATLON **	output	1	「6.2 L1、L2、L3共通データ」を参照してください。
from_scan	int	input	1	取得開始スキャン番号
to_scan	int	input	1	取得終了スキャン番号
access_lbl	int	input	1	「6.1.1.6 LATLON関数」を参照してください。

L1品質情報取得

L1データの品質情報を取得します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを取得します。
- データを格納するメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「sizeof(AM2_COMMON_LATLON)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Scan Data Quality (scan x 512) # データセットはchar型 x sizeof(AMTK_SCAN_DATA_QUALITY)サイズとなる
  : 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)

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
quality	AMTK_SCAN_DATA_QUALITY **	output	1	「6.2 L1、L2、L3共通データ」を参照してください。
from_scan	int	input	1	取得開始スキャン番号
to_scan	int	input	1	取得終了スキャン番号
access_lbl	int	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

整数型データ取得

HDFアクセスラベルを指定して整数型のデータを取得します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを取得します。
- データを格納するメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「sizeof(int)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Hot Load Count 6 to 36 (12 x scan x 16): int *p_data = malloc(sizeof(int) * 12 * scan * 16)
- データを格納するメモリエリア指定が、NULLの場合、関数内部でメモリエリアを確保します。

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	int **	output	1	データを格納するメモリエリア
from_scan	int	input	1	取得開始スキャン番号
to_scan	int	input	1	取得終了スキャン番号
access_lbl	int	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

実数型データ取得

HDFアクセスラベルを指定して実数型のデータを取得します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを取得します。
- データを格納するメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「sizeof(float)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Navigation Data (scan x 6): float *p_data = malloc(sizeof(float) * scan * 6)
- データを格納するメモリエリア指定が、NULLの場合、関数内部でメモリエリアを確保します。

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	float **	output	1	データを格納するメモリエリア
from_scan	int	input	1	取得開始スキャン番号
to_scan	int	input	1	取得終了スキャン番号
access_lbl	int	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

倍精度実数型データ取得

HDFアクセスラベルを指定して倍精度実数型のデータを取得します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを取得します。
- データを格納するメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「sizeof(double)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Position in Orbit (scan): double *p_data = malloc(sizeof(double) * scan)
- データを格納するメモリエリア指定が、NULLの場合、関数内部でメモリエリアを確保します。

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	double **	output	1	データを格納するメモリエリア
from_scan	int	input	1	取得開始スキャン番号
to_scan	int	input	1	取得終了スキャン番号
access_lbl	int	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

Unsigned Char 型データ取得

HDFアクセスラベルを指定してUnsigned Char型のデータを取得します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを取得します。
- データを格納するメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「sizeof(char)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  # sizeof(char)は1固定のため省略可
  例) SPC Temperature Count (scan x 34): unsigned char *p_data = malloc(scan * 34)
- データを格納するメモリエリア指定が、NULLの場合、関数内部でメモリエリアを確保します。

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	unsigned char **	output	1	データを格納するメモリエリア
from_scan	int	input	1	取得開始スキャン番号
to_scan	int	input	1	取得終了スキャン番号
access_lbl	int	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

(2) L1A、L1B、L1R、L2、L3 出力機能

プロダクトファイルオープン(書き込み専用)				
HDFプロダクトファイルを書き込み専用でオープンします。				
hid_t AMTK_openH5_Write(char *file_name, int mode)				
名前	型	入出力区別	サイズ	説明
戻り値				
HDF_file_id	hid_t	output	1	正常時:HDF access file id 異常時:オープンできない場合は、負の値が返ります。
パラメータ				
file_name	char *	input	1	AMSR2 HDFファイル名
mode	int	input	1	ファイルオープンモード AM2_CREATE_MODE : 新規作成 既存ファイルは削除されます。 AM2_RW_MODE : 上書き指定 既存ファイルがない場合、エラーとなります。

プロダクトファイルクローズ				
HDFプロダクトファイルをクローズします。				
int AMTK_closeH5_Write(hid_t HDF_file_id)				
名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:0 異常時:負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id

HDF データセット作成				
プロダクトへデータディメンジョンサイズのHDF データセットを作成します。 ー 生成する際、HDF データセットを欠損値で初期化します。				
int AMTK_setDimSize(hid_t HDF_file_id,int access_lbl , int *size)				
名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
access_lbl	int	input	1	「6.1.1 HDFアクセスラベル」を参照してください。
size	int *	input	1	ディメンジョンサイズ AMTK_getDimSizeの説明を参照してください。

メタデータ設定				
HDFプロダクトのメタデータ名でメタデータを設定します。				
int AMTK_setMetaDataName(hid_t HDF_file_id, char *name, char *value)				
名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
name	char *	input	1	メタデータ名
value	char *	input	1	メタデータ値

スキャン時刻設定

スキャン時刻(年、月、日、時、分、秒、ミリ秒)データをTAI93の実数型に変換し、指定されたスキャン番号に書き込みます。

- **スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。**
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを設定します。
- **設定するデータのメモリエリアは、関数を使用するユーザが確保します。**
  データサイズは、「sizeof(AM2_COMMON_SCANTIME)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) 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)

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	**正常時:0** **異常時:設定できない場合は、負の値が返ります。**
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
scan_time	AM2_COMMON_SCANTIME *	input	1	**「6.2 L1、L2、L3共通データ」を参照してください。**
from_scan	int	input	1	**設定開始スキャン番号**
to_scan	int	input	1	**設定終了スキャン番号**

緯度経度データ設定

緯度経度データを指定されたスキャン番号に書き込みます。

- **スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。**
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを設定します。
- **設定するデータのメモリエリアは、関数を使用するユーザが確保します。**
  データサイズは、「sizeof(AM2_COMMON_LATLON)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) 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)

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	**正常時:0** **異常時:設定できない場合は、負の値が返ります。**
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
latitudelongitude	AM2_COMMON_LATLON *	input	1	**「6.2 L1、L2、L3共通データ」を参照してください。**
from_scan	int	input	1	**設定開始スキャン番号**
to_scan	int	input	1	**設定終了スキャン番号**
access_lbl	int	input	1	**「6.1.1.6 LATLON関数」を参照してください。**

品質情報設定

L1データの品質情報を設定します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを設定します。
- 設定するデータのメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「sizeof(AM2_COMMON_LATLON)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Scan Data Quality (scan x 512) # データセットはchar型 x sizeof(AMTK_SCAN_DATA_QUALITY)サイズとなる
  : 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)

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
quality	AMTK_SCAN_DATA_QUALITY *	output	1	「6.2 L1、L2、L3共通データ」を参照してください。
from_scan	int	input	1	設定開始スキャン番号
to_scan	int	input	1	設定終了スキャン番号
access_lbl	int	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

整数型データ設定

HDFアクセスラベルを指定して整数型のデータとしてHDF データセットにデータを設定します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを設定します。
- 設定するデータのメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「sizeof(int)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Hot Load Count 6 to 36 (12 x scan x 16): int *p_data = malloc(sizeof(int) * 12 * scan * 16)
- HDF データセットに収容できない数値が設定データに存在した場合は、該当データをエラー値に置き換えHDFに書き込みます。また、戻り値statusにワーニングの負の値を設定します。

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	int *	input	1	設定するデータを格納するメモリエリア
from_scan	int	input	1	設定開始スキャン番号
to_scan	int	input	1	設定終了スキャン番号
access_lbl	int	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

実数型データ設定

HDFアクセスラベルを指定して実数型のデータとしてHDF データセットにデータを設定します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを設定します。
- 設定するデータのメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「sizeof(float)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Navigation Data (scan x 6): float *p_data = malloc(sizeof(float) * scan * 6)
- 実数の値を該当のHDF データセットのスケールファクタで、整数値へ変換する際は、スケールファクタで除算後、小数点第一位を四捨五入します。

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	float *	input	1	設定するデータを格納するメモリエリア
from_scan	int	input	1	設定開始スキャン番号
to_scan	int	input	1	設定終了スキャン番号
access_lbl	int	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

倍精度実数型データ設定

HDFアクセスラベルを指定して倍精度型のデータとしてHDF データセットにデータを設定します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを設定します。
- 設定するデータのメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「sizeof(double)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) 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)

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	double *	input	1	設定するデータを格納するメモリエリア
from_scan	int	input	1	設定開始スキャン番号
to_scan	int	input	1	設定終了スキャン番号
access_lbl	int	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

Unsigned Char 型データ設定

HDFアクセスラベルを指定してUnsigned Char型のデータとしてHDF データセットにデータを設定します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを取得します。
- データを格納するメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「sizeof(char)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  # sizeof(char)は1固定のため省略可
  例) 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)

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	unsigned char *	input	1	設定するデータを格納するメモリエリア
from_scan	int	input	1	設定開始スキャン番号
to_scan	int	input	1	設定終了スキャン番号
access_lbl	int	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

5.1.2 入力関数

(1) L1A、L1B、L1R 機能

共通関数を使用するため、L1 固有の入力関数はありません。

(2) L2 入力機能

共通関数を使用するため、L2 固有の入力関数はありません。

(3) L3 入力機能

整数型データ取得

HDFアクセスラベルを指定して整数型のデータを取得します。

- **アクセスラベルで指定したL3プロダクトの配列データの全てを取得します。**
- **取得するデータのメモリエリアは、関数を使用するユーザが確保します。**
  データサイズは、「sizeof(int)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Average Number (V) (Xpixel x Ypixel): int *p_data = malloc(sizeof(int) * Xpixel * Ypixel)
- **HDF データセットに収容できない数値が設定データに存在した場合は、該当データをエラー値に置き換えHDFに書き込みます。また、戻り値statusにワーニングの負の値を設定します。**
- **データを格納するメモリエリア指定が、NULLの場合、関数内部でメモリエリアを確保します。**

int AMTK_get_GridInt(hid_t HDF_file_id, int **data, int access_lbl)

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	int **	output	1	データを格納するメモリエリア
access_lbl	int	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

実数型データ取得

HDFアクセスラベルを指定して実数型のデータを取得します。

- **アクセスラベルで指定したL3プロダクトの配列データの全てを取得します。**
- **取得するデータのメモリエリアは、関数を使用するユーザが確保します。**
  データサイズは、「sizeof(float)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Standard Deviation (V) (Xpixel x Ypixel): float *p_data = malloc(sizeof(float) * Xpixel * Ypixel)
- **HDF データセットに収容できない数値が設定データに存在した場合は、該当データをエラー値に置き換えHDFに書き込みます。また、戻り値statusにワーニングの負の値を設定します。**
- **データを格納するメモリエリア指定が、NULLの場合、関数内部でメモリエリアを確保します。**

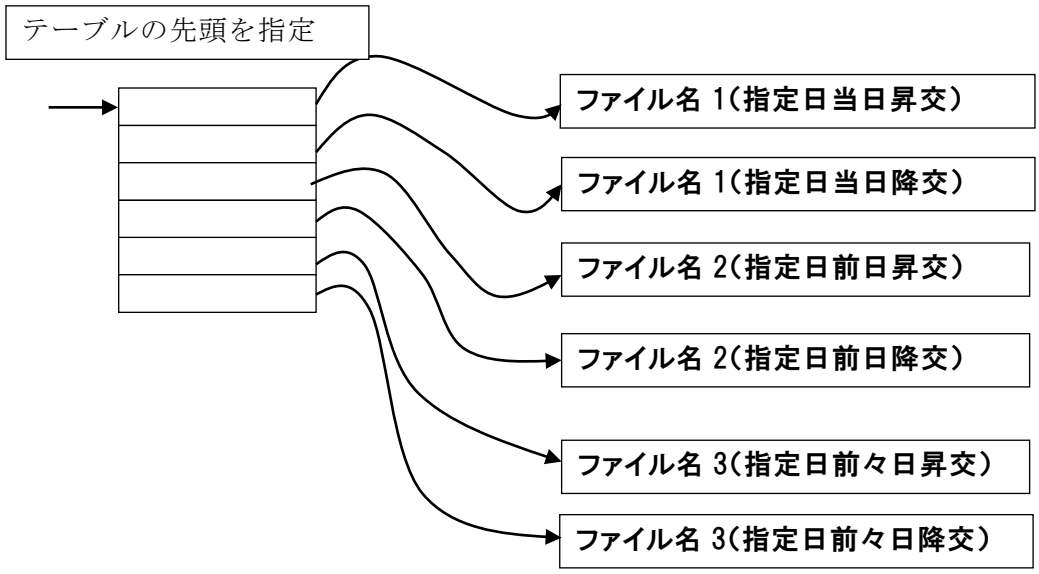
int AMTK_get_GridFloat(hid_t HDF_file_id, float **data, int access_lbl)

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id

data	float **	output	1	データを格納するメモリエリア
access_lbl	int	input	1	「6.1.1　HDFアクセスラベル」を参照してください。

3日平均データ取得				
レベル3の3日平均データを取得します。 入力ファイルが規定通りでない場合は、専用のエラー番号(-250～)が返却されます。 詳細は7　エラー番号を参照してください。				
int AMTK_get_Grid03D(float **average, char *name_table)				
名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:0 異常時:取得できない場合は、負の値が返ります。
パラメータ				
average	float **	output	M pixcel xN scan	データ構造体 0.1度格子、0.25度格子は、ファイル名で判断します。 指定するファイルは、全て同じ格子のファイルを指定します。
name_table	char *	input	char [6][512]	ファイル名テーブル(下図を参照してください。)

ファイル名テーブル (char[6][512])



5.1.3 出力関数

(1) L1 出力機能

共通関数を使用するため、L1 固有の出力関数はありません。

(2) L2 出力機能

共通関数を使用するため、L2 固有の出力関数はありません。

(3) L3 出力機能

整数型データ設定				
HDFアクセスラベルを指定して整数型のデータとしてHDF データセットにデータを設定します。 - アクセスラベルで指定したL3プロダクトの配列データの全てを設定します。 配列サイズの情報は、AMTK_getDimSize関数を使用します。 - 設定するデータのメモリエリアは、関数を使用するユーザが確保します。 データサイズは、「sizeof(int)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。 例) Average Number (V) (Xpixel x Ypixel): int *p_data = malloc(sizeof(int) * Xpixel * Ypixel)				
int AMTK_set_GridInt(hid_t HDF_file_id, int *data, int access_lbl)				
名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	int *	input	1	設定するデータを格納するメモリエリア
access_lbl	int	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

実数型データ設定				
HDFアクセスラベルを指定して実数型のデータとしてHDF データセットにデータを設定します。 - アクセスラベルで指定したL3プロダクトの配列データの全てを設定します。 配列サイズの情報は、AMTK_getDimSize関数を使用します。 - 設定するデータのメモリエリアは、関数を使用するユーザが確保します。 データサイズは、「sizeof(float)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。 例) Standard Deviation (V) (Xpixel x Ypixel): float *p_data = malloc(sizeof(float) * Xpixel * Ypixel)				
int AMTK_set_GridFloat(hid_t HDF_file_id, float *data, int access_lbl)				
名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	float *	input	1	設定するデータを格納するメモリエリア
access_lbl	int	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

5.2 Fortran 言語

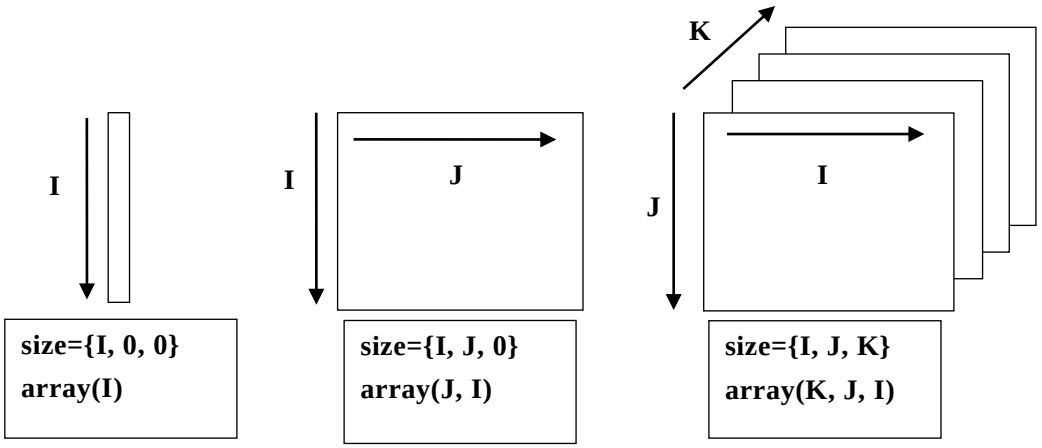
5.2.1 共通関数

(1) L1A、L1B、L1R、L2、L3 入力機能

プロダクトファイルオープン(読込み専用)				
HDFプロダクトファイルを読込み専用でオープンします。				
hid_t = AMTK_openH5(file_name)				
名前	型	入出力区別	サイズ	説明
戻り値				
HDF_file_id	integer	output	1	正常時:HDF access file idが返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
file_name	character	input	1	AMSR2 HDFファイル名

プロダクトファイルクローズ				
HDFプロダクトファイルをクローズします。				
status = AMTK_closeH5(HDF_file_id)				
名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:0 異常時:負の値が返ります。
パラメータ				
HDF_file_id	integer	input	1	HDFアクセスfile id

データディメンジョンサイズ取得				
HDFプロダクトからデータディメンジョンサイズを取得します。 ー 複数のデータセットを対象とするアクセルラベルが指定された場合は、エラーを返します。				
status = AMTK_getDimSize(HDF_file_id, access_lbl , size)				
名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時：配列の次元数 異常時：取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
access_lbl	integer	input	1	「6.1.1　HDFアクセスラベル」を参照してください。
size(3)	integer	output	1	ディメンジョンサイズ （サイズと配列の関係は下図を参照してください。配列に関しては、C言語の次元の並びと逆順になります。）



一次元データ

スキャン数: **I**

二次元データ

スキャン数: **I**

三次元データ

スキャン数: **J**

「6.1.2　物理量に関するデータセットの説明」も併せて参照してください。

メタデータ取得(メタデータ名)

HDFプロダクトのメタデータ名でメタデータを取得します。
　ー メタデータ値を設定するメモリ領域がNULLの場合は、エラーを返します。

status = AMTK_getMetaDataName(HDF_file_id, name, value)

名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:メタデータ文字数 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	integer	input	1	HDFアクセスfile id
name	character	input	1	メタデータ名
value	character	output	1	メタデータ値

メタデータ取得(インデックス値)

HDFプロダクトのインデックス値でメタデータを取得します。
　ー メタデータ値を設定するメモリ領域がNULLの場合は、エラーを返します。

status = AMTK_getMetaData(HDF_file_id, index, value)

名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:メタデータ文字数 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	integer	input	1	HDFアクセスfile id
index	integer	input	1	メタデータインデックス値
value	character	output	1	メタデータ値

スキャン時刻取得

スキャン時刻を取得します。
- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF　データセットに格納されているスキャン番号より大きい場合は、HDF　データセットに格納されているデータまでを取得します。
- データを格納するメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「24(=AM2_COMMON_SCANTIMEのサイズ)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。例) Scan Time (scan): type(AM2_COMMON_SCANTIME) data(scan)

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
from_scan	integer	input	1	取得開始スキャン番号
to_scan	integer	input	1	取得終了スキャン番号
scan_time	AM2_COMMON_SCANTIME	output	1	「6.2　L1、L2、L3共通データ」を参照してください。

緯度経度データ取得

指定されたスキャン番号の範囲の緯度経度データをHDFから取得します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを取得します。
- データを格納するメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「8(=AM2_COMMON_LATLONのサイズ)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) 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)

名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
latitudelongitude	AM2_COMMON_LATLON *	output	1	「6.2 L1、L2、L3共通データ」を参照してください。
from_scan	integer	input	1	取得開始スキャン番号
to_scan	integer	input	1	取得終了スキャン番号
access_lbl	integer	input	1	「6.1.1.6 LATLON関数」を参照してください。

L1品質情報取得

L1データの品質情報を取得します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを取得します。
- データを格納するメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「512(=AM2_COMMON_LATLONのサイズ)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Scan Data Quality (scan x 512) # データセットはchar型 x sizeof(AMTK_SCAN_DATA_QUALITY)サイズとなる
  : type(AMTK_SCAN_DATA_QUALITY) data(scan)

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
quality	AMTK_SCAN_DATA_QUALITY *	output	1	「6.2 L1、L2、L3共通データ」を参照してください。
from_scan	int	input	1	取得開始スキャン番号
to_scan	int	input	1	取得終了スキャン番号
access_lbl	int	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

整数型データ取得

HDFアクセスラベルを指定して整数型のデータを取得します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを取得します。
- データを格納するメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「4(＝integerのサイズ*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Hot Load Count 6 to 36 (12 x scan x 16): integer data(16 * scan * 12)
- データを格納するメモリエリア指定が、NULLの場合、関数内部でメモリエリアを確保します。

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	integer	output	1	データを格納するメモリエリア
from_scan	integer	input	1	取得開始スキャン番号
to_scan	integer	input	1	取得終了スキャン番号
access_lbl	integer	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

実数型データ取得

HDFアクセスラベルを指定して実数型のデータを取得します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを取得します。
- データを格納するメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「4(=realのサイズ)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Navigation Data (scan x 6): real data(6, scan)
- データを格納するメモリエリア指定が、NULLの場合、関数内部でメモリエリアを確保します。

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	real	output	1	データを格納するメモリエリア
from_scan	integer	input	1	取得開始スキャン番号
to_scan	integer	input	1	取得終了スキャン番号
access_lbl	integer	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

倍精度実数型データ取得

HDFアクセスラベルを指定して倍精度実数型のデータを取得します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを取得します。
- データを格納するメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「8(real*8のサイズ)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Position in Orbit (scan): real*8 data(scan)

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	real*8	output	1	データを格納するメモリエリア
from_scan	integer	input	1	取得開始スキャン番号
to_scan	integer	input	1	取得終了スキャン番号
access_lbl	integer	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

Character 型データ取得

HDFアクセスラベルを指定してCharacter型のデータを取得します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを取得します。
- データを格納するメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「1(=characterのサイズ)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) SPC Temperature Count (scan x 34): character data(34, scan)
- データを格納するメモリエリア指定が、NULLの場合、関数内部でメモリエリアを確保します。

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	Character	output	1	データを格納するメモリエリア
from_scan	integer	input	1	取得開始スキャン番号
to_scan	integer	input	1	取得終了スキャン番号
access_lbl	integer	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

(2) L1A、L1B、L1R、L2、L3 出力機能

プロダクトファイルオープン(書き込み専用)				
HDFプロダクトファイルを書き込み専用でオープンします。				
hid_t = AMTK_openH5_Write(file_name, mode)				
名前	型	入出力区別	サイズ	説明
戻り値				
HDF_file_id	integer	output	1	正常時:HDF access file id 異常時:取得できない場合は、負の値が返ります。
パラメータ				
file_name	character	input	1	AMSR2 HDFファイル名
mode	integer	input	1	ファイルオープンモード AM2_CREATE_MODE : 新規作成 既存ファイルは削除されます。 AM2_RW_MODE : 上書き指定 既存ファイルがない場合、エラーとなります。

プロダクトファイルクローズ				
HDFプロダクトファイルをクローズします。				
status = AMTK_closeH5_Write(HDF_file_id)				
名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:0 異常時:負の値が返ります。
パラメータ				
HDF_file_id	integer	input	1	HDFアクセスfile id

HDF データセット作成				
HDFプロダクトへデータディメンジョンサイズのHDF データセットを作成します。 ー 生成する際、HDF データセットを初期化します。				
status = AMTK_setDimSize(HDF_file_id, access_lbl, size)				
名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
access_lbl	integer	input	1	「6.1.1 HDFアクセスラベル」を参照してください。
size(3)	integer	input	1	ディメンジョンサイズ（三次元配列。解説は、AMTK_getDimSizeの関数説明を参照してください。）

メタデータ設定				
HDFプロダクトのメタデータ名でメタデータを設定します。				
status = AMTK_setMetaDataName(HDF_file_id, name, value)				
名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	integer	input	1	HDFアクセスfile id
name	character	input	1	メタデータ名
value	character	input	1	メタデータ値

スキャン時刻設定

スキャン時刻(年、月、日、時、分、秒、ミリ秒)データをTAI93の実数型に変換し、指定されたスキャン番号の位置に書き込みます。

- **スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。**
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを設定します。
- **設定するデータのメモリエリアは、関数を使用するユーザが確保します。**
  データサイズは、「24(=AM2_COMMON_SCANTIMEのサイズ)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Scan Time (scan): type(AM2_COMMON_SCANTIME) data(scan)

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	**正常時:0** **異常時:設定できない場合は、負の値が返ります。**
パラメータ				
HDF_file_id	integer	input	1	HDFアクセスfile id
scan_time	AM2_COMMON_SCANTIME	input	1	**「6.2 L1、L2、L3共通データ」を参照してください。**
from_scan	integer	input	1	**設定開始スキャン番号**
to_scan	integer	input	1	**設定終了スキャン番号**

緯度経度データ設定

緯度経度データを指定されたスキャン番号の範囲のデータをHDFに書き込みます。

- **スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。**
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを設定します。
- **設定するデータのメモリエリアは、関数を使用するユーザが確保します。**
  データサイズは、「8(=AM2_COMMON_LATLONのサイズ)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) 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)

名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	**正常時:0** **異常時:設定できない場合は、負の値が返ります。**
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
latitudelongitude	AM2_COMMON_LATLON	input	1	**「6.2 L1、L2、L3共通データ」を参照してください。**
from_scan	integer	input	1	**設定開始スキャン番号**
to_scan	integer	input	1	**設定終了スキャン番号**
access_lbl	integer	input	1	**「6.1.1 HDFアクセスラベル」を参照してください。**

L1品質情報設定

L1データの品質情報を設定します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを設定します。
- 設定するデータのメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「512(=AM2_COMMON_LATLONのサイズ)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Scan Data Quality (scan x 512) # データセットはchar型 x sizeof(AMTK_SCAN_DATA_QUALITY)サイズとなる
  : type(AMTK_SCAN_DATA_QUALITY) data(scan)

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	int	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
quality	AMTK_SCAN_DATA_QUALITY	output	1	「6.2 L1、L2、L3共通データ」を参照してください。
from_scan	int	input	1	設定開始スキャン番号
to_scan	int	input	1	設定終了スキャン番号
access_lbl	int	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

整数型データ設定

HDFアクセスラベルを指定して整数型のデータとしてHDF データセットにデータを設定します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを設定します。
- 設定するデータのメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「4(=integerのサイズ*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Hot Load Count 6 to 36 (12 x scan x 16): integer data(16 * scan * 12)
- HDF データセットに収容できない数値が設定データに存在した場合は、該当データをエラー値に置き換えHDFに書き込みます。また、戻り値statusにワーニングの負の値を設定します。

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	integer	input	1	設定するデータを格納するメモリエリア
from_scan	integer	input	1	設定開始スキャン番号
to_scan	integer	input	1	設定終了スキャン番号
access_lbl	integer	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

実数型データ設定

HDFアクセスラベルを指定して実数型のデータとしてHDF データセットにデータを設定します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを設定します。
- 設定するデータのメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「4(=realのサイズ)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Navigation Data (scan x 6): real data(6, scan)
- 実数の値を該当のHDF データセットのスケールファクタで、整数値へ変換する際は、スケールファクタで除算後、小数点第一位を四捨五入します。

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	real	input	1	設定するデータを格納するメモリエリア
from_scan	integer	input	1	設定開始スキャン番号
to_scan	integer	input	1	設定終了スキャン番号
access_lbl	integer	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

倍精度実数型データ設定

HDFアクセスラベルを指定して倍精度型のデータとしてHDF データセットにデータを設定します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを設定します。
- 設定するデータのメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「8(real*8のサイズ)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Position in Orbit (scan): real*8 data(scan)

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	real*8	input	1	設定するデータを格納するメモリエリア
from_scan	integer	input	1	設定開始スキャン番号
to_scan	integer	input	1	設定終了スキャン番号
access_lbl	integer	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

Character 型データ設定

HDFアクセスラベルを指定してCharacter型のデータとしてHDF データセットにデータを設定します。

- スキャン番号は、開始スキャン番号と終了スキャン番号を指定します。
  終了スキャン番号が、開始スキャン番号より小さい場合は、エラーが返ります。
  終了スキャン番号が指定されたHDF データセットに格納されているスキャン番号より大きい場合は、HDF データセットに格納されているデータまでを設定します。
- 設定するデータのメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「1(=characterのサイズ)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) SPC Temperature Count (scan x 34): character data(34, scan)

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

名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	character	input	1	設定するデータを格納するメモリエリア
from_scan	integer	input	1	設定開始スキャン番号
to_scan	integer	input	1	設定終了スキャン番号
access_lbl	integer	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

5.2.2 入力関数

(1) L1A、L1B、L1R 機能

共通関数を使用するため、L1 固有の出力関数はありません。

(2) L2 入力機能

共通関数を使用するため、L2 固有の出力関数はありません。

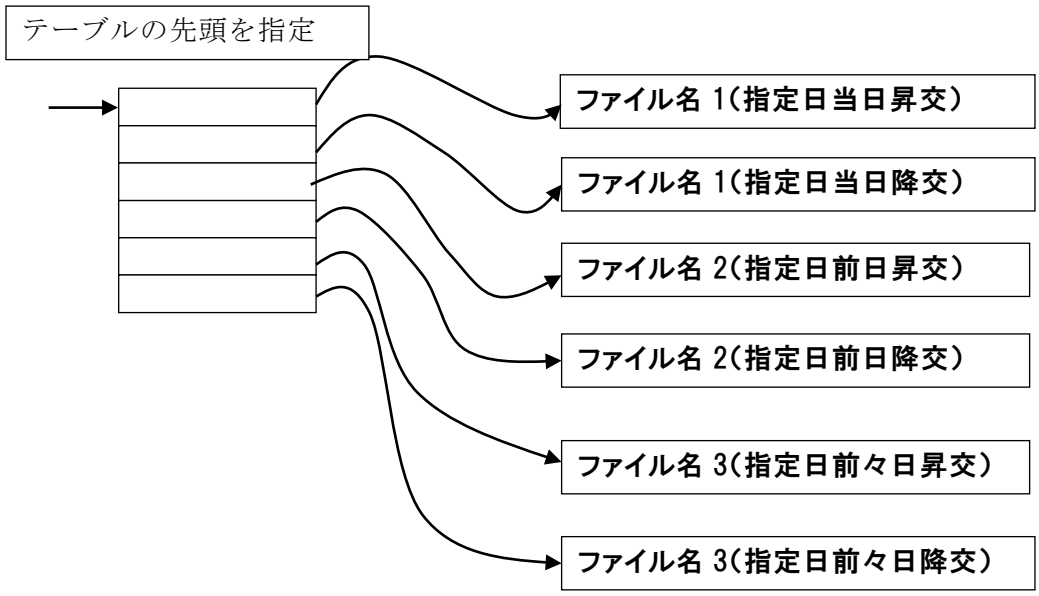
(3) L3 入力機能

整数型のデータ取得				
HDFアクセスラベルを指定して整数型のデータを取得します。 - アクセスラベルで指定したL3プロダクトの配列データの全てを取得します。 配列サイズの情報は、AMTK_getDimSize関数を使用します。 - 取得するデータのメモリエリアは、関数を使用するユーザが確保します。 データサイズは、「4(=integerのサイズ)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。 例) Average Number (V) (Xpixel x Ypixel): integer data(Ypixel * Xpixel)				
status = AMTK_get_GridInt(HDF_file_id, data, access_lbl)				
名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	integer	output	1	データを格納するメモリエリア
access_lbl	integer	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

実数型のデータ取得				
HDFアクセスラベルを指定して実数型のデータを取り出します。 - アクセスラベルで指定したL3プロダクトの配列データの全てを取得します。 配列サイズの情報は、AMTK_getDimSize関数を使用します。 - 取得するデータのメモリエリアは、関数を使用するユーザが確保します。 データサイズは、「4(=realのサイズ)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。 例) Standard Deviation (V) (Xpixel x Ypixel): real data(Ypixel * Xpixel)				
status = AMTK_get_GridFloat(HDF_file_id, data, access_lbl)				
名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:取得スキャン数が返ります。 異常時:取得できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	real	output	1	データを格納するメモリエリア
access_lbl	integer	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

3日平均データ取得				
レベル3の3日平均データを取得します。 入力ファイルが規定通りでない場合、専用のエラー番号(-250～)が返却されます。 詳細は7　エラー番号を参照してください。				
status = AMTK_get_Grid03D(average, name_table)				
名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:0 異常時:取得できない場合は、負の値が返ります。
パラメータ				
average	real	output	M pixcel xN scan	データ構造体 0.1度格子、0.25度格子は、ファイル名で判断します。 指定するファイルは、全て同じ格子のファイルを指定します。
name_table	character*512(6)	input	character *512(6)	ファイル名テーブル(下図を参照してください。)

ファイル名テーブル (character*512(6)))



5.2.3 出力関数

(1) L1 出力機能

共通関数を使用するため、L1 固有の出力関数はありません。

(2) L2 出力機能

共通関数を使用するため、L2 固有の出力関数はありません。

(3) L3 出力機能

整数型のデータ設定

HDFアクセスラベルを指定して整数型のデータとしてHDF データセットにデータを設定します。
- アクセスラベルで指定したL3プロダクトの配列データの全てを設定します。
  配列サイズの情報は、AMTK_getDimSize関数を使用します。
- 設定するデータのメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「4(=integerのサイズ)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Average Number (V) (Xpixel x Ypixel): integer data(Ypixel * Xpixel)

status = AMTK_set_GridInt(HDF_file_id, data, access_lbl)

名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	integer	input	1	設定するデータを格納するメモリエリア
access_lbl	integer	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

実数型のデータ設定

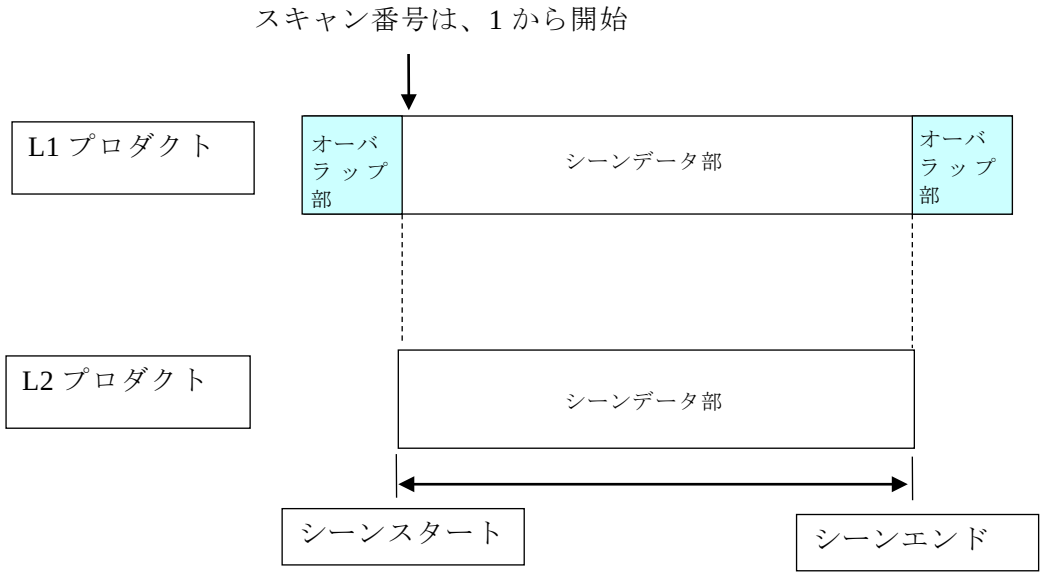
HDFアクセスラベルを指定して実数型のデータとしてHDF データセットにデータを設定します。
- アクセスラベルで指定したL3プロダクトの配列データの全てを設定します。
  配列サイズの情報は、AMTK_getDimSize関数を使用します。
- 設定するデータのメモリエリアは、関数を使用するユーザが確保します。
  データサイズは、「4(=realのサイズ)*ディメンジョンサイズ(AMTK_getDimSize関数参照)」とします。
  例) Standard Deviation (V) (Xpixel x Ypixel): real data(Ypixel * Xpixel)

status = AMTK_set_GridFloat(HDF_file_id, data, access_lbl)

名前	型	入出力区別	サイズ	説明
戻り値				
status	integer	output	1	正常時:0 異常時:設定できない場合は、負の値が返ります。
パラメータ				
HDF_file_id	hid_t	input	1	HDFアクセスfile id
data	real	input	1	設定するデータを格納するメモリエリア
access_lbl	integer	input	1	「6.1.1 HDFアクセスラベル」を参照してください。

5.3　スキャン番号

AMTKの関数は、スキャン番号を指定して入出力を行います。スキャン番号は、L1プロダクト、L2プロダクト共に１から開始されます。L1プロダクトは、オーバラップ部を含むため、図のようにL1とL2のシーンデータ部を一致させるようにします。



L1プロダクトでオーバラップ部をアクセスする場合は、以下のように指定します。

シーンスタートより前の値：マイナス値（オーバラップ）～０の範囲

シーンエンドより後ろの値：シーンエンド＋オーバラップ

シーンデータ部は、メタデータのオーバーラップスキャン数(OverlapScans)、シーンスキャン数（NumberOfScans）の数値で、シーンスタートとシーンエンドを定義します。

シーンスタート：オーバーラップスキャン数＋１の位置

シーンエンド　：オーバーラップスキャン数＋シーンスキャン数

全てのスキャンを対象とする場合、スキャン数の定義を以下のようにします。

シーンスキャン数　　：オーバーラップを含まないスキャン数

プロダクトスキャン数：シーンスキャン数＋オーバラップスキャン数×２

L2は、シーンスキャン数とプロダクトスキャン数が一致しますが、L1は、オーバラップ部があるため一致しません。

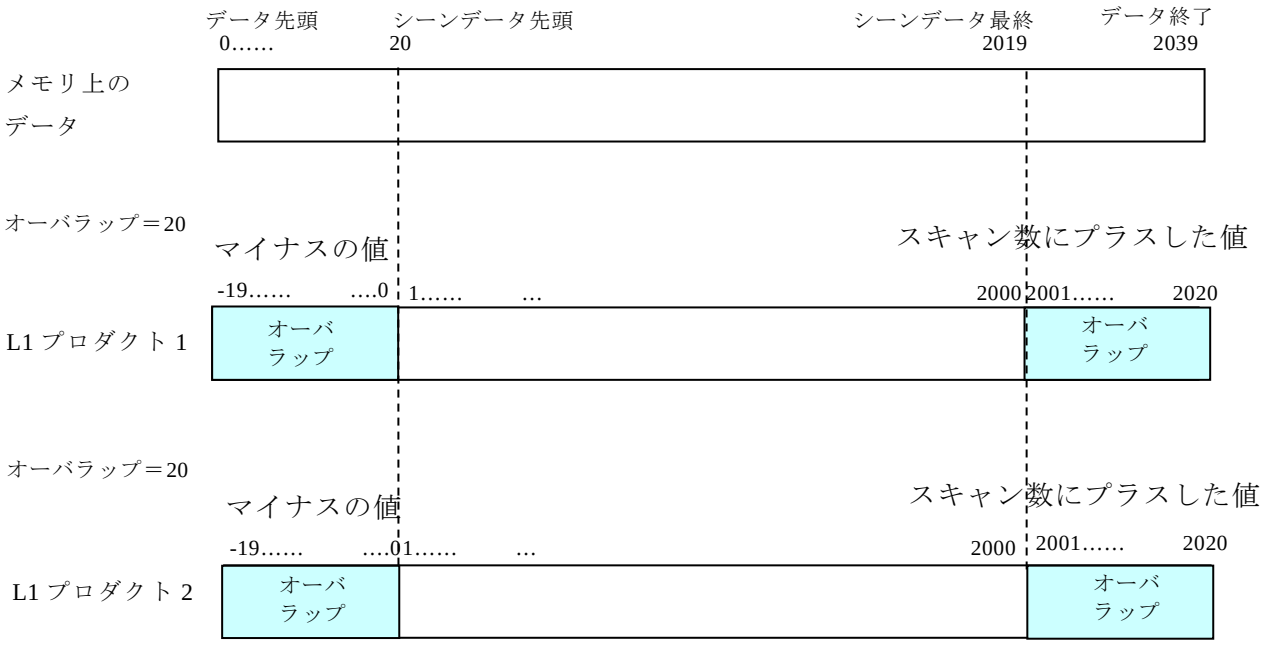
AMTKの関数へのスキャン番号指定は、プロダクトスキャン数の範囲を超えた場合はエラーとなります。

オーバーラップスキャン数が同一のL1入力→L1出力のケースと、オーバーラップスキャン数が異なるL1入力→L2出力のケースの指定例を次ページに示します。

入力L1・出力L1

読み込み：-19～2020と指定

書き込み：-19～2020と指定　　メモリ上のデータ位置は、データ先頭

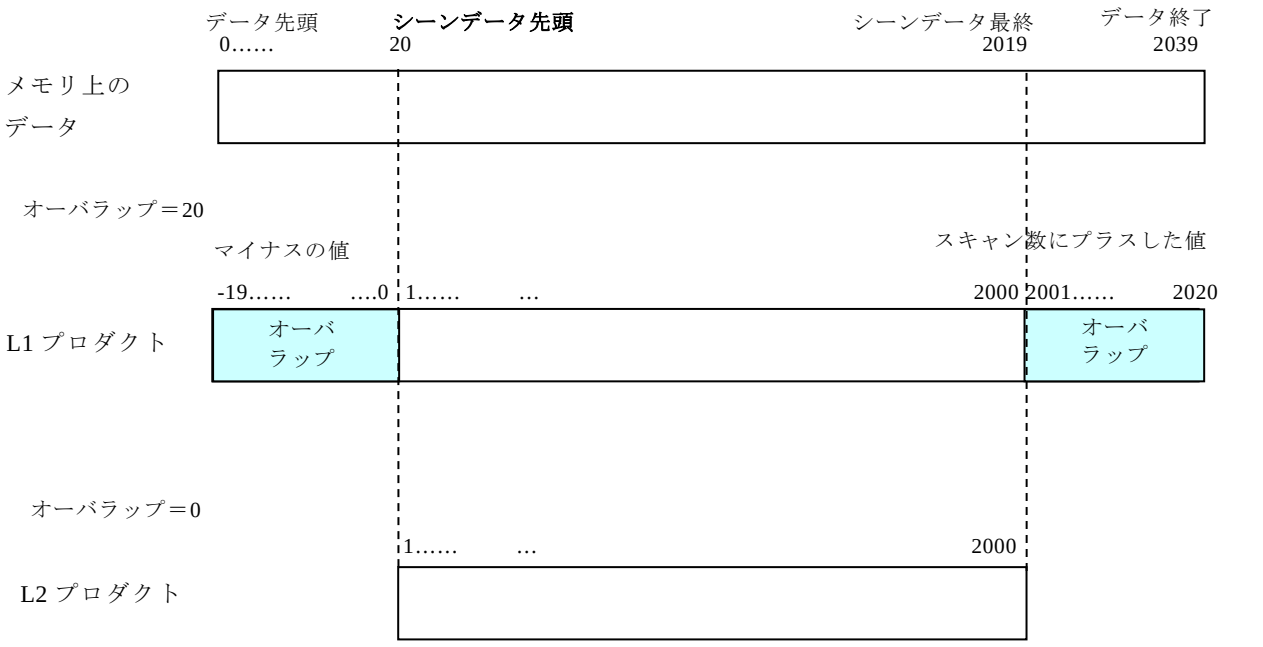


入力L1・出力L2

読み込み：-19～2020と指定

書き込み：1～2000と指定、

書き込むメモリのデータ位置は、シーンデータ先頭を指定します。



(注)プロダクトのオーバラップスキャン数は、メタデータ「OverlapScans」の値を参照しています。前述の例は、各プロダクトが以下の値であることを前提としています。

・L1プロダクト ： OverlapScans = 20

・L2プロダクト ： OverlapScans = 0

5.4　格納される値について

AMTKの関数は、データの型により分けられています。また、HDFに格納されるデータもデータの型がありますが、関数の型とHDFに格納されるデータの型が、一致しない場合もあります。

関数の型とHDFに格納されるデータの型により、設定される値が異なる場合がありますので注意してください。ここでは、データセットが最初に作成される時にHDFに格納される値と入出力の際に格納される値について示します。

データセットを作成するには、AMTK_setDimSize関数が使用されます。この関数は、指定されたサイズでデータセットを作成し、下表に示す格納値を初期値としてHDFに書き込みます。

C 言語の型	Fortran の型	格納値	備考
Int	Integer*4	-32766	
unsigned int	-	65533	
float	real*4	-9997.0	
double	real*8	-9997.0	
unsihned char	-	253	

入出力の際に格納される値は、使用する関数とプロダクトのデータの型は、組み合わせで決まります。また、使用するデータには、欠損値と異常値があります。欠損値は、指定された場所にデータが存在しない場合に設定します。異常値は、パリティエラーなど値の異常を示す場合に設定します。欠損値、異常値のどちらも決まった値となるようスケールファクタの処理は行いません。関数とプロダクトのデータの型の組み合わせと欠損値、異常値を以下に示します。各欄の上段が、欠損値、下段が、異常値を示します。データの型は、C言語の型で示します。

C 言語の型	Int()	Float()	Double()	UChar	備考
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　入出力データ

6.1　データ定義

6.1.1　HDF アクセスラベル

AMTKの関数は、汎用的に作られているため、HDFのアクセス単位であるデータセットを特定するために各々のデータセットに番号が付与されています。ユーザは、直接数字を指定するか、表に示すアクセスラベルとして定義された識別子を使用します。

アクセスラベルを指定してデータセットを入出力する際、AMTKの関数内で以下のメタデータを参照します。あらかじめメタデータに適切な値を格納してください。これらのメタデータが存在しない場合は、各メタデータごとに固有のエラー番号を返却します。

詳細は7　エラー番号を参照してください。

メタデータ	参照内容
ProductName	プロダクト種別を判別するために参照します。
OverlapScans	入出力対象のスキャン番号を特定するために参照します。(スキャン番号については「5.3　スキャン番号」を参照してください) プロダクト種別がL3の場合は、参照しません。
GeophysicalName	L2, L3の場合、物理量を判別するために参照します。(物理量については「2.5　物理量定義ファイルについて」を参照してください)
GranuleID	L3の場合、プロダクト種別を判別する為に参照します。
CoRegistrationParameterA1	以下のアクセスラベルを指定したとき、低周波緯度経度を算出する為に参照します。 AM2_LATLON_06, AM2_LATLON_07, AM2_LATLON_10, AM2_LATLON_18, AM2_LATLON_23, AM2_LATLON_36, AM2_LATLON_LO
CoRegistrationParameterA2	
Projection	アクセスラベルAM2_GRID_LATLONを指定したとき、地図投影法を判別する為に参照します。
OrbitDirection	AMTK_get_Grid03D()を実行したとき、軌道方向を判別する為に参照します。
ProductionDateTime	AMTK_get_Grid03D()を実行したとき、プロダクト生成時を取得する為に参照します。

次ページ以降の表の内容は、以下の内容が示されています。

データ名	HDF ファイルのデータセット名です。
型	HDFに格納されるデータ型と、アクセス関数の引数データ型です。
アクセス関数	アクセス関数には、入力用と出力用があり、目的に合わせて関数を使用します。「－」があるものは、対象の関数が使用できないことを示します。HDFに格納されている型とアクセス関数は異なる場合があることにご注意ください。データセットに整数値のデータが格納されていても、Scale Facterを考慮するため実数値のアクセス関数になる場合があります。
アクセスラベル	HDF データセットにアクセスするための識別子です。(アクセスIDの別名定義)
アクセスID	HDF データセットにアクセスするための番号です。
備考	Scale Facterの値など参考情報です。

6.1.1.1　L1A

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
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	

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
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	
25	Hot Load Count 6 to 36	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_HTS_LO	10240	3 次元データセットの入出力方法は、6.1.2.1 の説明を参照してください。
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	

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
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	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	
31	Longitude of Observation Point for 89A	float	float	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	
32	Latitude of Observation Point for 89B	float	float	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	
33	Longitude of Observation Point for 89B	float	float	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	
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	3 次元データセットの入出力方法は、6.1.2.1 の説明を参照してください。
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	unsigned char	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	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PCD	10450	

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
44	Scan Data Quality	binary (512byte) = unsigned char*512	AMTK_SCAN_DATA_QUALITY	AMTK_getScanDataQuality	AMTK_setScanDataQuality	AM2_SCAN_QUAL	10460	。
45	Pixel Data Quality 6 to 36	binary (2byte) = unsigned char*2	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_LO	10470	3 次元データセットの入出力方法は、6.1.2.1 の説明を参照してください。 データの格納方法は、6.1.2.2 を参照してください。
46	Pixel Data Quality 89	binary (1byte) = unsigned char*1	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_HI	10480	
47	Interpolation Flag 6 to 36	binary (1byte) = unsigned char*1	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_INTPL_LO	10490	3 次元データセットの入出力方法は、6.1.2.1 の説明を参照してください。
48	Interpolation Flag 89	binary (1byte) = unsigned char*1	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_INTPL_HI	10500	

6.1.1.1 L1B

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
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	

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
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	
25	Hot Load Count 6 to 36	signed int	signed int	AMTK_get_SwathInt	AMTK_set_SwathInt	AM2_HTS_LO	10240	3 次元データセットの入出力方法は、6.1.2.1 の説明を参照してください。
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	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	
31	Longitude of Observation Point for 89A	float	float	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
32	Latitude of Observation Point for 89B	float	float	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	
33	Longitude of Observation Point for 89B	float	float	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	
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	3 次元データセットの入出力方法は、6.1.2.1 の説明を参照してください。
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	unsigned char	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	unsigned char	AMTK_set_SwathUChar	AMTK_set_SwathUChar	AM2_PCD	10450	
44	Scan Data Quality	binary (512byte) = unsigned char*512	AMTK_SCAN_DATA_QUALITY	AMTK_getScanDataQuality	AMTK_setScanDataQuality	AM2_SCAN_QUAL	10460	

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
45	Pixel Data Quality 6 to 36	binary (2byte) = unsigned char*2	unsigned char	AMTK_set_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_LO	10470	3 次元データセットの入出力方法は、6.1.2.1 の説明を参照してください。データの格納方法は、6.1.2.2 を参照してください。
46	Pixel Data Quality 89	binary (1byte) = unsigned char*1	unsigned char	AMTK_set_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_HI	10480	
47	Interpolation Flag 6 to 36	binary (1byte) = unsigned char*1	unsigned char	AMTK_set_SwathUChar	AMTK_set_SwathUChar	AM2_INTPL_LO	10490	3 次元データセットの入出力方法は、6.1.2.1 の説明を参照してください。
48	Interpolation Flag 89	binary (1byte) = unsigned char*1	unsigned char	AMTK_set_SwathUChar	AMTK_set_SwathUChar	AM2_INTPL_HI	10500	

6.1.1.2 L1R

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
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	

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
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 (res06,89.0GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES06_TB89H	12180	
20	Brightness Temperature (res06,all)	unsigned int	float	AMTK_get_SwathFloat	ー	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	

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
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 (res23,89.0GHz,H)	unsigned int	float	AMTK_get_SwathFloat	AMTK_set_SwathFloat	AM2_RES23_TB89H	12380	
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	

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
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	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	
53	Longitude of Observation Point for 89A	float	float	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	
54	Latitude of Observation Point for 89B	float	float	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	
55	Longitude of Observation Point for 89B	float	float	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
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	
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	3 次元データセットの入出力方法は、6.1.2.1 の説明を参照してください。
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) = unsigned char*512	AMTK_SCAN_DATA_QUALITY	AMTK_getScanDataQuality	AMTK_setScanDataQuality	AM2_SCAN_QUAL	10460	
64	Pixel Data Quality 6 to 36	binary (2byte) = unsigned char*2	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_LO	10470	3 次元データセットの入出力方法は、6.1.2.1 の説明を参照してください。データの格納方法は、6.1.2.2 を参照してください。
65	Pixel Data Quality 89	binary (1byte) = unsigned char*1	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_HI	10480	

6.1.1.3　L2

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
低解像度								
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	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。
5						AM2_SWATH_GEO2	21040	
6						AM2_SWATH_GEO3	21050	
7						AM2_SWATH_GEOA	21060	
8	Latitude of Observation Point	float	float	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	
9	Longitude of Observation Point	float	float	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	
10	Pixel Data Quality	binary (1byte) = unsigned char*1	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL	21070	3 次元データセットの入出力方法は、6.1.2.1 の説明を参照してください。 データの格納方法は、6.1.2.2 を参照してください。

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
高解像度								
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	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。
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	
9						AM2_SWATHB_GEO2	22080	
10						AM2_SWATHB_GEO3	22090	
11						AM2_SWATHB_GEOA	22100	
12	Latitude of Observation Point for 89A	float	float	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	
13	Longitude of Observation Point for 89A	float	float	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	
14	Latitude of Observation Point for 89B	float	float	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
15	Longitude of Observation Point for 89B	float	float	LATLON 関数を参照	LATLON 関数を参照	LATLON 関数を参照	–	
16	Pixel Data Quality for 89A	binary (1byte) = unsigned char*1	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_A	22110	3 次元データセットの入出力方法は、6.1.2.1 の説明を参照してください。 データの格納方法は、6.1.2.2 を参照してください。
17	Pixel Data Quality for 89B	binary (1byte) = unsigned char*1	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL_B	22120	

注１　物理量ごとのスケールファクタを以下の表に示します。
物理量の定義はユーザによって追加や変更が可能です。「2.5　物理量定義ファイルについて」を参照して下さい。

物理量		scale factor
名称	メタデータ名	
積算水蒸気量	Total Precipitable Water	0.01
積算雲水量	Cloud Liquid Water	0.001
海上風	Sea Surface Wind speed	0.01
海面温度	Sea Surface Temperature	0.01
海氷密接度	Sea Ice Concentration	0.1
積雪	Snow Depth	0.1
土壌水分	Soil Moisture Content	0.1
降水量	Precipitation,	0.01

6.1.1.4　L3

日単位（高解像度）

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
EQR，輝度温度								
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	Time Information	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TIME	31040	
PS 北半球，輝度温度								
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	Time Information	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TIME	31040	
PS 南半球，輝度温度								
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	Time Information	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_TIME	31040	

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
EQR, 物理量								
1	ProductMeta Data	–					–	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。
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 北半球, 海氷密接度								
1	ProductMeta Data	–					–	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。
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 南半球, 海氷密接度								
1	ProductMeta Data	–					–	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。
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	

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
PS 北半球，積雪深								
1	ProductMeta Data	–					–	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。
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	

日単位（低解像度）

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
EQR，輝度温度								
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 北半球，輝度温度								
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 南半球，輝度温度								
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	

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
EQR, 物理量								
1	ProductMeta Data	–					–	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。。
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 北半球, 海氷密接度								
1	ProductMeta Data	–					–	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。
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 南半球, 海氷密接度								
1	ProductMeta Data	–					–	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。
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	

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
PS 北半球, 積雪深								
1	ProductMeta Data	–					–	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。
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	

月単位（高解像度）

	データ名	型(HDF 格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
EQR, 輝度温度								
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	

	データ名	型(HDF 格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセス ID	備考
				入力	出力			
PS 北半球, 輝度温度								
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	

	データ名	型(HDF 格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセス ID	備考
				入力	出力			
PS 南半球, 輝度温度								
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	

	データ名	型(HDF 格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセス ID	備考
				入力	出力			
EQR, 物理量								
1	ProductMeta Data	–					–	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。
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	
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	
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	
15						AM2_GRID_GEO2_TNUM	34600	
16						AM2_GRID_GEO3_TNUM	34610	
17						AM2_GRID_GEOA_TNUM	34620	

	データ名	型(HDF 格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセス ID	備考
				入力	出力			
PS 北半球, 海氷密接度								
1	ProductMeta Data	–					–	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。。
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	
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	
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	
15						AM2_GRID_GEO2_TNUM	34600	
16						AM2_GRID_GEO3_TNUM	34610	
17						AM2_GRID_GEOA_TNUM	34620	

	データ名	型(HDF 格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセス ID	備考
				入力	出力			
PS 南半球, 海氷密接度								
1	ProductMeta Data	–					–	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。。
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	
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	
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	
15						AM2_GRID_GEO2_TNUM	34600	
16						AM2_GRID_GEO3_TNUM	34610	
17						AM2_GRID_GEOA_TNUM	34620	

	データ名	型(HDF 格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセス ID	備考
				入力	出力			
PS 北半球, 積雪深								
1	ProductMeta Data	–					–	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。。
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	
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	
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	
15						AM2_GRID_GEO2_TNUM	34600	
16						AM2_GRID_GEO3_TNUM	34610	
17						AM2_GRID_GEOA_TNUM	34620	

月単位（低解像度）

	データ名	型(HDF 格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
EQR, 輝度温度								
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	

	データ名	型(HDF 格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセス ID	備考
				入力	出力			
PS 北半球, 輝度温度								
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	

	データ名	型(HDF 格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセス ID	備考
				入力	出力			
PS 南半球, 輝度温度								
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	

	データ名	型(HDF 格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセス ID	備考
				入力	出力			
EQR, 物理量								
1	ProductMeta Data	–					–	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。
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	
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	
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	
15						AM2_GRID_GEO2_TNUM	34600	
16						AM2_GRID_GEO3_TNUM	34610	
17						AM2_GRID_GEOA_TNUM	34620	

	データ名	型(HDF 格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセス ID	備考
				入力	出力			
PS 北半球，海氷密接度								
1	ProductMeta Data	–					–	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。
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	
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	
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	
15						AM2_GRID_GEO2_TNUM	34600	
16						AM2_GRID_GEO3_TNUM	34610	
17						AM2_GRID_GEOA_TNUM	34620	

	データ名	型(HDF 格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセス ID	備考
				入力	出力			
PS 南半球，海氷密接度								
1	ProductMeta Data	–					–	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。
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	
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	
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	
15						AM2_GRID_GEO2_TNUM	34600	
16						AM2_GRID_GEO3_TNUM	34610	
17						AM2_GRID_GEOA_TNUM	34620	

	データ名	型(HDF 格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセス ID	備考
				入力	出力			
PS 北半球，積雪深								
1	ProductMeta Data	–					–	
2	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GEO1	31310	入出力方法が他のデータセットと異なります。6.1.2.1 の説明を参照してください。
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	
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	
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	
15						AM2_GRID_GEO2_TNUM	34600	
16						AM2_GRID_GEO3_TNUM	34610	
17						AM2_GRID_GEOA_TNUM	34620	

6.1.1.5　LATLON 関数

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
1	LatitudeLongitude (L1A/L1B 6.9GHz)	float	AM2_COMMON_LATLON	AMTK_getLatLon	－	AM2_LATLON_06	40010	89A の緯度経度から相対レジストレーションを使用して各チャンネルの緯度経度を返します。
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	Latitude of Observation Point for 89A Longitude of Observation Point for 89A	float	AM2_COMMON_LATLON	AMTK_getLatLon	AMTK_setLatLon	AM2_LATLON_89A	40070	HDF 格納値
8	Latitude of Observation Point for 89B Longitude of Observation Point for 89B	float	AM2_COMMON_LATLON	AMTK_getLatLon	AMTK_setLatLon	AM2_LATLON_89B	40080	HDF 格納値
9	LatitudeLongitude (L1A/L1B Low Mean)	float	AM2_COMMON_LATLON	AMTK_getLatLon	－	AM2_LATLON_LO	40090	Low(06-36GHz)の緯度・経度の平均値を返します。

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
10	Latitude of Observation Point for 89A Longitude of Observation Point for 89A	float	AM2_COMMON_LATLON	AMTK_getLatLon	AMTK_setLatLon	AM2_LATLON_RS_89A	40100	
11	Latitude of Observation Point for 89B Longitude of Observation Point for 89B	float	AM2_COMMON_LATLON	AMTK_getLatLon	AMTK_setLatLon	AM2_LATLON_RS_89B	40110	
12	LatitudeLongitude（L1R リサンプリング）	float	AM2_COMMON_LATLON	AMTK_getLatLon	－	AM2_LATLON_RS_LO	40120	高度補正済みResampling pointを返します。 ※AM2_LATLON_RS_89Aの奇数点(C言語配列上は0, 2, 4...)
13	Latitude of Observation Point Longitude of Observation Point	float	AM2_COMMON_LATLON	AMTK_getLatLon	AMTK_setLatLon	AM2_LATLON_L2_LO	40130	L2緯度・経度を返します。
14	Latitude of Observation Point for 89A Longitude of Observation Point for 89A	float	AM2_COMMON_LATLON	AMTK_getLatLon	AMTK_setLatLon	AM2_LATLON_L2_89A	40140	
15	Latitude of Observation Point for 89B Longitude of Observation Point for 89B	float	AM2_COMMON_LATLON	AMTK_getLatLon	AMTK_setLatLon	AM2_LATLON_L2_89B	40150	

	データ名	型(HDF格納)	型(関数引数)	アクセス関数		アクセスラベル	アクセスID	備考
				入力	出力			
16	LatitudeLongitude（L3 日単位/月単位 高解像度 EQR）	float	AM2_COMMON_LATLON	AMTK_getLatLon	－	AM2_GRID_LATLON	40160	環境変数L3LATLONFILEDIRで指定された緯度経度ファイルを参照し、該当の緯度経度を返します。開始スキャン番号、終了スキャン番号の指定は、無効となります。
17	LatitudeLongitude（L3 日単位/月単位 高解像度 PS 北半球）	float	AM2_COMMON_LATLON	AMTK_getLatLon	－	AM2_GRID_LATLON	40160	
18	LatitudeLongitude（L3 日単位/月単位 高解像度 PS 南半球）	float	AM2_COMMON_LATLON	AMTK_getLatLon	－	AM2_GRID_LATLON	40160	
19	LatitudeLongitude（L3 日単位/月単位 低解像度 EQR）	float	AM2_COMMON_LATLON	AMTK_getLatLon	－	AM2_GRID_LATLON	40160	
20	LatitudeLongitude（L3 日単位/月単位 低解像度 PS 北半球）	float	AM2_COMMON_LATLON	AMTK_getLatLon	－	AM2_GRID_LATLON	40160	
21	LatitudeLongitude（L3 日単位/月単位 低解像度 PS 南半球）	float	AM2_COMMON_LATLON	AMTK_getLatLon	－	AM2_GRID_LATLON	40160	

6.1.2　物理量に関するデータセットの説明

6.1.2.1　3 次元データセットの入出力方法

AMTKは、3次元データセットを「通常の3次元データセット」と「物理量に関する3次元データセット」の2種類に分類し、それぞれ異なる入出力方法を提供しています。両者は、データセットのサイズを入出力するときの「スキャン数を表す次元」が異なる特徴を持ちます。

表 6.1-1に、それぞれの入出力方法を示します。

表 6.1-1 3 次元データセットの入出力方法一覧

No.	種別	入出力の仕様
1	通常の3次元データセット	AMTK_setDimSize()、AMTK_getDimSize()にてデータセットのサイズを入出力するとき、ディメンジョンサイズのスキャン数は**2 次元目**とする。
		例) Hot Load Count 6 to 36のサイズ: チャンネル数 * スキャン数 * ピクセル数 dimsize[0] = 12 dimsize[1] = 2000　// 2次元目がスキャン数 dimsize[2] = 16
2	物理量に関する 3 次元データセット （2次元データを1～3層持ち、各層へ個別にアクセスするデータセット）	AMTK_setDimSize()、AMTK_getDimSize()にてデータセットのサイズを入出力するとき、ディメンジョンサイズのスキャン数は**1 次元目**とし、3 次元目を 1～3 層のレイヤーとする。
		例) Geopysical Dataのサイズ: スキャン数 * ピクセル数 * レイヤー数 dimsize[0] = 2000　// 1次元目がスキャン数 dimsize[1] = 243 dimsize[2] = 3　　// 3次元目がレイヤー数

6.1.2.1.1 通常の 3 次元データセット一覧

○L1

No.	データセット名	HDF 格納型	入出力型	入力関数	出力関数	アクセスラベル	アクセス 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.	データセット名	HDF 格納型	入出力型	入力関数	出力関数	アクセスラベル	アクセス ID
低解像度							
1	Pixel Data Quality	binary (1byte) = unsigned char*1	unsigned char	AMTK_get_SwathUChar	AMTK_set_SwathUChar	AM2_PIX_QUAL	21070
高解像度							
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 物理量に関する 3 次元データセット一覧

・アクセスラベルのAM2_*Aは、3次元データセット（Scan * Pixel * Layerサイズ）として入出力します。
・アクセスラベルのAM2_*1～*3は、1～3個目のレイヤーのデータを2次元データセット（Scan * Pixelサイズ）として入出力します。

○L2

No.	データセット名	HDF 格納型	入出力型	入力関数	出力関数	アクセスラベル	アクセス 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.	データセット名	HDF 格納型	入出力型	入力関数	出力関数	アクセスラベル	アクセス ID
1	Geophysical Data	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GE01	31310
2						AM2_GRID_GE02	31320
3						AM2_GRID_GE03	31330
4						AM2_GRID_GE0A	31340
5	Standard Deviation	signed int	float	AMTK_get_GridFloat	AMTK_set_GridFloat	AM2_GRID_GE01_STD	34510
6						AM2_GRID_GE02_STD	34520
7						AM2_GRID_GE03_STD	34530
8						AM2_GRID_GE0A_STD	34540
9	Average Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GE01_ANUM	34550
10						AM2_GRID_GE02_ANUM	34560
11						AM2_GRID_GE03_ANUM	34570
12						AM2_GRID_GE0A_ANUM	34580
13	Total Number	signed int	signed int	AMTK_get_GridInt	AMTK_set_GridInt	AM2_GRID_GE01_TNUM	34590
14						AM2_GRID_GE02_TNUM	34600
15						AM2_GRID_GE03_TNUM	34610
16						AM2_GRID_GE0A_TNUM	34620

6.1.2.1.3 3 次元データセットのスキャン数を表す次元

通常の 3 次元データセットは、2 次元目をスキャン数と扱います。一方、物理量に関する 3 次元データセットは、1 次元目をスキャン数と扱います。図 6-1に、両者のディメンジョンサイズにおけるスキャン数を表す次元の差異を示します。

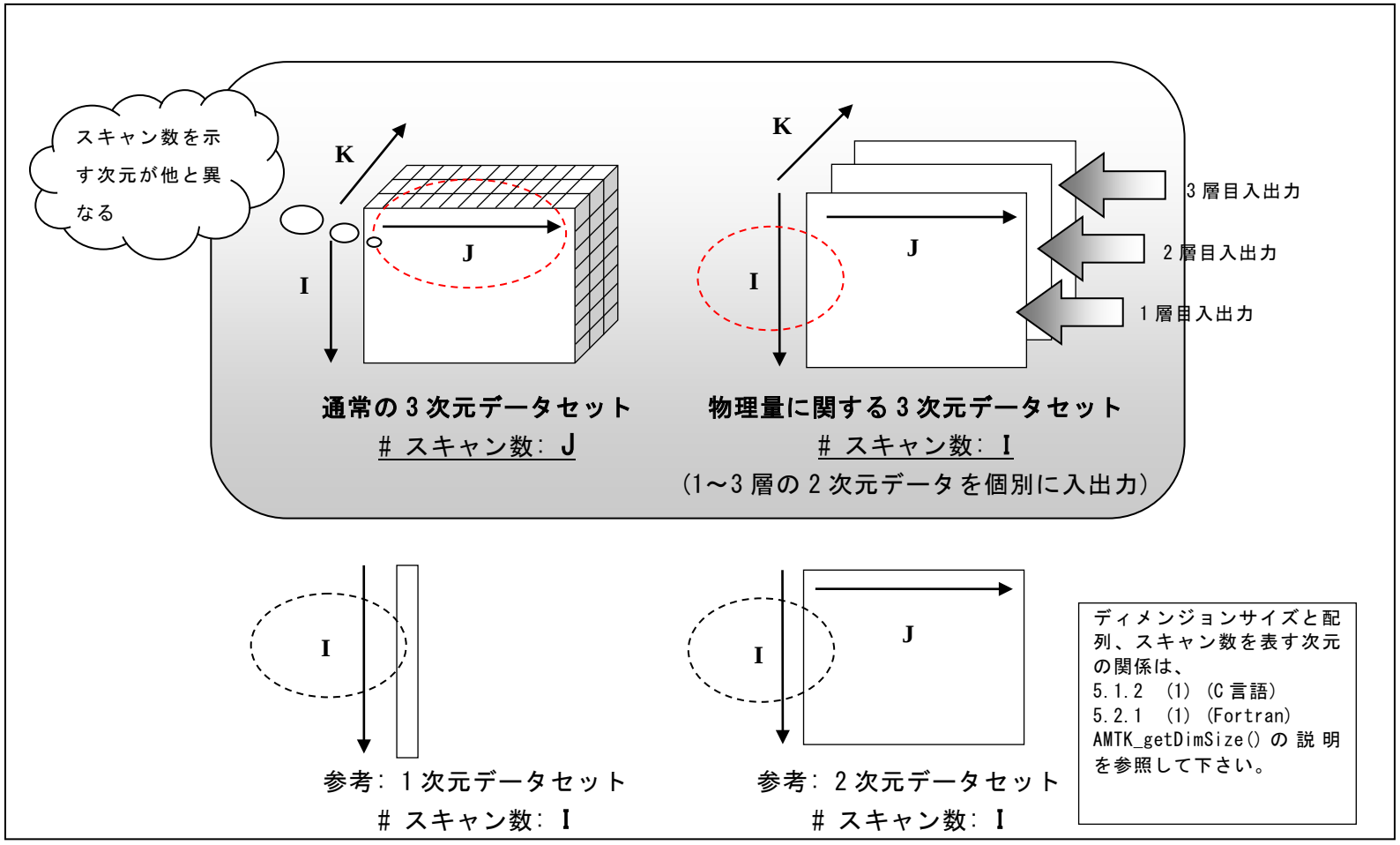


図 6-1 ディメンジョンサイズにおけるスキャン数を表す次元の差異

物理量に関する3次元データセットであるL2、L3のGeophysical Dataデータセット、L3のStandard Deviation、Average Number、Total Numberデータセットは、2次元データを1～3層持つ構造となっています。AMTKは、全層を3次元データとして一括で入出力するアクセスラベルと、1層目、2層目、3層目の2次元データを個別に入出力するアクセスラベルの2種類を提供します。図 6-2にイメージを示します。

・一括アクセスラベル：AM2_* _GEOA*（全層に対する3次元データアクセス）
サイズ入出力関数AMTK_setDimSize()、AMTK_getDimSize()に使用します。

・個別アクセスラベル：AM2_* _GEO1*、AM2_* _GEO2*、AM2_* _GEO3* （各層に対する2次元データアクセス）
データ入出力関数AMTK_set_Swath*()、AMTK_get_Swath*()、AMTK_set_Grid*()、AMTK_get_Grid*()に使用します。

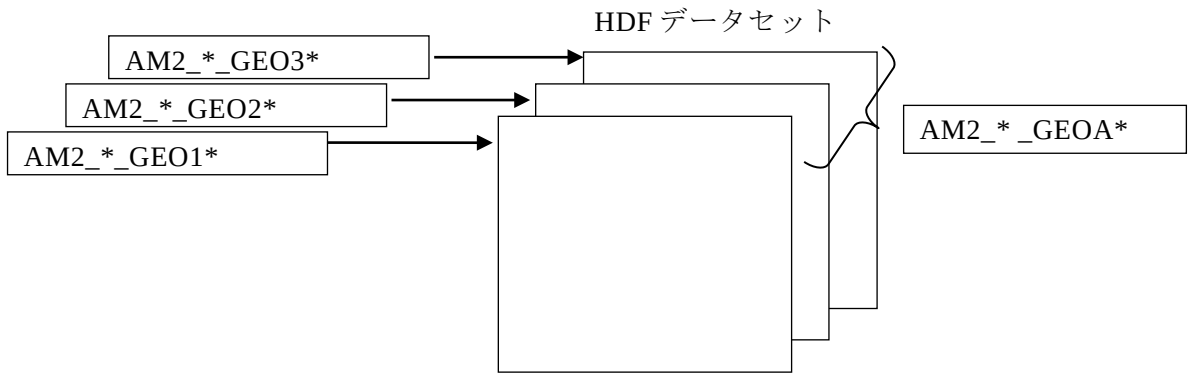


図 6-2　2 次元データ個別アクセスラベルと 3 次元データー括アクセスラベル

通常の3次元データセット、物理量に関する3次元データセットの具体的な入出力例は、L2、L3の入出力サンプルプログラムを参考にしてください。以下に、L2の入出力サンプルプログラムの一部を抜粋して説明します。

(1) 通常の 3 次元データセットの出力例

sample3_make_L2Lproduct.c : Pixel Data Quality データセット出力（抜粋）

```
...

/** 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);
          }
}
```

●変数の宣言
この例では、出力する Pixel Data Quality データセットの物理量(レイヤー)数を 3 とします。

●メモリ領域の確保
出力するデータセットの領域を確保します。

malloc()にてレイヤー数 * スキャン数 * ピクセル数サイズのメモリ領域を確保します。

●データセットの作成
出力するディメンジョンサイズを設定します。
このとき、2 次元目にスキャン数を指定します。
1 次元目: レイヤー数
2 次元目: スキャン数
3 次元目: ピクセル数

AM2_PIX_QUAL アクセスラベルを使用して、レイヤー数 * スキャン数 * ピクセル数のサイズの 3 次元データセットを出力します。

●データの書き込み
出力するデータセットの値を設定します。

(2) 物理量に関する 3 次元データセットの出力例

sample3_make_L2Lproduct.c :Geophysical Data データセット出力（抜粋）

```
...

/** 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++)
        {
```

●変数の宣言
この例では、出力する Geophysical Data データセットの物理量(レイヤー)数を 3 とします。

Geophysial Data データセットの 1～3 層へのアクセスラベルを宣言します。

●メモリ領域の確保
出力するデータセットの領域を確保します。

malloc にてスキャン数 * ピクセル数サイズのメモリ領域を、計 3 層分確保します。

●データセットの作成
出力するディメンジョンサイズを設定します。
このとき、1 次元目にスキャン数を指定します。
1 次元目：スキャン数
2 次元目：ピクセル数
3 次元目：レイヤー数

AM2_SWATH_GEOA アクセスラベルを使用して、スキャン数 * ピクセル数 * レイヤー数のサイズの 3 次元データセットを出力します。

●データの書き込み
出力するデータセットの値を設定します。

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);
terminate(file_id, &dataset);
exit(EXIT_FAILURE);
}
}
}
...
	Geophysial Data データセットの 1～3 層へのアクセスラベルを使用して、スキャン数 ＊ピクセル数の 2 次元データセットの値を計 3 層分出力します。

6.1.2.2　Pixel Data Quality データセット格納方法

AMTK を使用して Pixel Data Quality データセットを入出力するとき、C 言語は unsigned char 型、Fortran は character 型の配列を使用し、各要素の bit に設定すべき値を格納します。

L1 の Pixel Data Quality 6 to 36 は 16bit 単位(配列サイズ=2)、Pixel Data Quality 89 は 8bit 単位(配列サイズ=1)で意味のある値となります。配列の各要素の bit に設定すべき値を、図 6-3に示します。

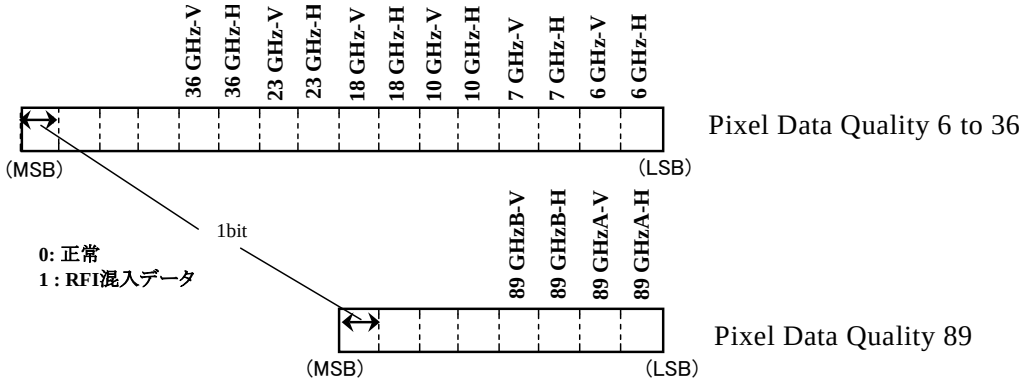


図 6-3　L1 の Pixel Data Quality 6 to 36, 89 の格納イメージ

尚、HDF ファイルへの入出力は 1byte 単位で行われます。

L1 の Pixel Data Quality 6 to 36、Pixel Data Quality 89 のデータセットと配列の対応イメージを、図 6-4に示します。

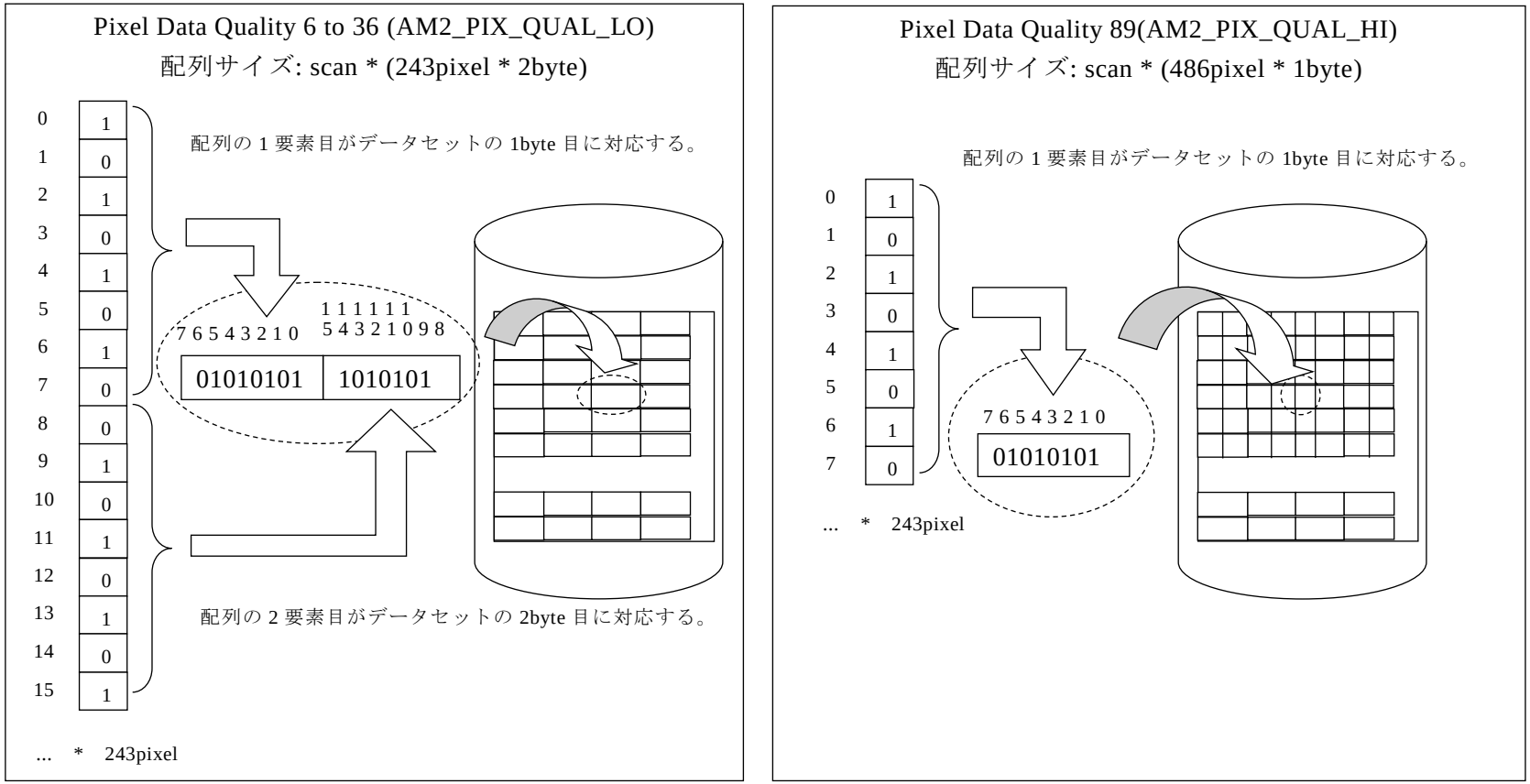


図 6-4　L1 の Pixel Data Quality 6 to 36, 89 のデータセットと配列の対応イメージ

6.1.3 プロダクト関連

プロダクトに関するデータ定義を表に示します。

	名前	値	説明	備考
サンプル数	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	1080	Number of pixels in Y direction for L3 High-res PN (SND)	
	AM2_DEF_L3H_PN2_Y	1435	Number of pixels in X direction for L3 High-res PN (SND)	
	AM2_DEF_L3L_PN2_X	432	Number of pixels in Y direction for L3 Low-res PN (SND)	
	AM2_DEF_L3L_PN2_Y	574	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	
校正情報	AM2_DEF_CAL_LO	16	CAL data sampling number(AMSR-E 16)	
	AM2_DEF_CAL_HI	32	CAL data sampling number(AMSR-E 32)	

	名前	値	説明	備考
欠損値	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作成モード	AM2_RW_MODE	1	AMTK_openH5_Write() : read/write mode	
	AM2_CREATE_MODE	2	AMTK_openH5_Write() : create mode	

6.2　L1、L2、L3 共通データ

プロダクト共通のデータ構造体を以下に示します。

対応データ		AMSR2のデータ構造体	備考
	スキャンタイム	AM2_COMMON_SCANTIME	
	緯度経度	AM2_COMMON_LATLON	
	品質データ	AMTK_SCAN_DATA_QUALITY	L1で使用

スキャンタイム			
AM2_COMMON_SCANTIME			
名前	型	サイズ	説明
tai93sec	double	1	1993年1月1日0時0分00秒からの通算秒
year	short	1	年
month	short	1	月
day	short	1	日
hour	short	1	時
minute	short	1	分
second	short	1	秒
ms	short	1	ミリ秒

緯度経度			
AM2_COMMON_LATLON			
名前	型	サイズ	説明
lat	float	1	緯度[deg]
lon	float	1	経度[deg]

品質データ			
AMTK_SCAN_DATA_QUALITY			
名前	型	サイズ	説明
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　エラー番号

AMTKの関数で異常時のエラー番号を表 7-1エラー番号一覧に示します。
番号は、マイナス値で示し、以下のカテゴリで番号が付与されています。

-100番台　　プログラム実行環境のエラー
-200番台　　AMTKを使用するユーザプログラムバグに起因するエラー
-500番台　　HDFファイルに関係するエラー
-600番台　　設定データに関するエラー

表 7-6.2-1エラー番号一覧

番号	説明	備考
-100	うるう秒ファイルなどのファイルのオープンができません。	
-101	メモリ領域の確保ができません。	
-102	環境変数の取得ができません。	
-103	うるう秒の取得ができません。	
-109	物理量定義ファイルを取得できません。ファイルが存在しない､もしくはファイルの定義内容が誤っています。	
-201	スキャン時刻の指定年が 1993 年以前です。	
-202	スキャン時刻の指定月が範囲外です。	月の指定は、1 から 12 です。
-203	スキャン時刻の指定日が範囲外です。	日の指定は、1 から 31 です。
-204	スキャン時刻の指定時が範囲外です。	時の指定は、1 から 24 です。
-205	スキャン時刻の指定分が範囲外です。	分の指定は、0 から 59 です。
-206	スキャン時刻の指定秒が範囲外です。	秒の指定は、0 から 60 です。
-207	スキャン時刻の指定ミリ秒が範囲外です。	ミリ秒の指定は、0 から 999 です。
-208	スキャン時刻の指定が範囲外です。	
-209	指定された通算秒がマイナス値です。	
-210	ファイル名の指定がありませんでした。	
-211	終了スキャン番号より開始スキャン番号の方が大きな値です。	
-213	HDF ファイル ID が 0 かマイナス値です。	
-214	メタデータのインデックス値がマイナス値です。	
-215	指定されたディメンジョンサイズが不正です。	
-216	書き込みデータの指定がありません。	
-217	アクセスラベルが 0 で指定されました。	
-218	指定アクセスラベルが書込対象外のものです。	
-219	指定アクセスラベルが読込対象外です。	
-220	メモリ領域を取得しようとした際にデータサイズが 0 です。	
-221	パラメータで指定されたデータのアドレスが NULL です。	
-238	HDF ファイル書き込みモードの指定が誤っています。	
-239	アクセスラベルと対応する関数のデータ型が異なっています。	

番号	説明	備考
-250	物理量が規定と異なるため、3 日平均データを取得できません。 物理量(メタデータ"GeophysicalName")は、平均を取得するファイル全て同一である必要があります。	参考: 5　AMTK の関数 3 日平均データ取得 AMTK_get_Grid03D()
-251	軌道方向が規定と異なるため、3 日平均データを取得できません。 起動方向(メタデータ"OrbitDirection")は、奇数ファイルは Ascending、偶数ファイルは Descending を設定する必要があります。	参考: 5　AMTK の関数 3 日平均データ取得 AMTK_get_Grid03D()
-252	プロダクト生成時が規定と異なるため、3 日平均データを取得できません。 プロダクト生成時(メタデータ"ProductionDateTime")の年月日は、奇数ファイルと偶数ファイルて同一である必要があります。また、ファイル名 1 を指定日とし、ファイル名 2 は指定日前日、ファイル名 3 は指定日前々日となっている必要があります。	参考: 5　AMTK の関数 3 日平均データ取得 AMTK_get_Grid03D()
-253	物理量のサイズが規定と異なるため、3 日平均データを取得できません。 物理量(データセット"GeophysicalData")のサイズは、平均を取得するファイル全て同一である必要があります。	参考: 5　AMTK の関数 3 日平均データ取得 AMTK_get_Grid03D()
-501	HDF ファイルオープンができません。	
-502	HDF ファイルクローズができません。	
-503	ディメンジョンサイズの設定ができません。	
-504	データセットアクセスに失敗しました。	
-505	データセットのオープンができません。	
-506	データセットのスペース取得ができません。	
-507	データセット配列次元数の取得ができません。	
-508	データセットのディメンジョンサイズ取得ができません。	
-509	データセットの作成ができません。	
-510	データセットへ書き込みができません。	
-511	データセットの読み込みができません。	
-520	メタデータ名のオープンができません。	
-521	メタデータ名のデータ種別の取得ができません。	
-522	メタデータ値の読み込ができません。	
-523	メタデータ名のインデックス指定で、オープンができません。	
-524	メタデータ名の作成ができません。	
-525	メタデータ値の書き込みができません。	
-526	データ種別クラス取得ができません。	
-527	データ種別のコピーができません。	
-528	データ種別のサイズ設定ができません。	
-529	データ種別の取得ができません。	
-540	データ種別のスペース取得ができません。	
-541	データセット読み書き位置指定ができま	

番号	説明	備考
	せん。	
-550	メタデータ"ProductName"オープンエラーのため、データセットのアクセスに失敗しました。	参考: 6.1.1　HDF アクセスラベル
-551	メタデータ"ProductName"の読み込み時にHDF ライブラリ内部エラーが発生し、データセットのアクセスに失敗しました。	参考: 6.1.1　HDF アクセスラベル
-552	メタデータ"OverlapScans"オープンエラーのため、データセットのアクセスに失敗しました。	参考: 6.1.1　HDF アクセスラベル
-553	メタデータ"OverlapScans"の読み込み時に HDF ライブラリ内部エラーが発生し、データセットのアクセスに失敗しました。	参考: 6.1.1　HDF アクセスラベル
-554	メタデータ"GeophysicalName"オープンエラーのため、データセットのアクセスに失敗しました。	参考: 6.1.1　HDF アクセスラベル
-555	メタデータ"GeophysicalName"の読み込み時に HDF ライブラリ内部エラーが発生し、データセットのアクセスに失敗しました。	参考: 6.1.1　HDF アクセスラベル
-556	メタデータ"GranuleID"オープンエラーのため、データセットのアクセスに失敗しました。	参考: 6.1.1　HDF アクセスラベル
-557	メタデータ"GranuleID"の読み込み時にHDF ライブラリ内部エラーが発生し、データセットのアクセスに失敗しました。	参考: 6.1.1　HDF アクセスラベル
-558	メタデータ"CoRegistrationParameterA1"オープンエラーのため、データセットのアクセスに失敗しました。	参考: 6.1.1　HDF アクセスラベル
-559	メタデータ"CoRegistrationParameterA1"の読み込み時に HDF ライブラリ内部エラーが発生し、データセットのアクセスに失敗しました。	参考: 6.1.1　HDF アクセスラベル
-560	メタデータ"CoRegistrationParameterA2"オープンエラーのため、データセットのアクセスに失敗しました。	参考: 6.1.1　HDF アクセスラベル
-561	メタデータ"CoRegistrationParameterA2"の読み込み時に HDF ライブラリ内部エラーが発生し、データセットのアクセスに失敗しました。	参考: 6.1.1　HDF アクセスラベル
-562	メタデータ"Projection"オープンエラーのため、データセットのアクセスに失敗しました。	参考: 6.1.1　HDF アクセスラベル
-563	メタデータ"Projection"の読み込み時にHDF ライブラリ内部エラーが発生し、データセットのアクセスに失敗しました。	参考: 6.1.1　HDF アクセスラベル
-564	メタデータ"OrbitDirection"オープンエラーのため、データセットのアクセスに失敗しました。	参考: 6.1.1　HDF アクセスラベル
-565	メタデータ"OrbitDirection"の読み込み時に HDF ライブラリ内部エラーが発生し、データセットのアクセスに失敗しました。	参考: 6.1.1　HDF アクセスラベル
-566	メタデータ"ProductionDateTime"オープンエラーのため、データセットのアクセスに失敗しました。	参考: 6.1.1　HDF アクセスラベル
-567	メタデータ"ProductionDateTime"の読み	参考: 6.1.1　HDF アクセスラベル

番号	説明	備考
	込み時に HDF ライブラリ内部エラーが発生し､データセットのアクセスに失敗しました。	
-601	読み書きするデータ種別がデータセットのデータ種別と一致していません。	
-602	データ書き込み時の最大最小値が異常です。	
9999	書き込みデータが最大値を超えました。	