第一期水循環変動観測衛星「しずく」(GCOM-W1)搭載
高性能マイクロ波放射計 2(AMSR2)データ利用解説文書

第四版　平成 28 年 11 月 17 日

宇宙航空研究開発機構(JAXA)
地球観測研究センター(EORC)

改訂履歴

版	日付	改訂内容
第一版	平成 25 年 2 月 1 日	第一版発行
第二版	平成 25 年 3 月 29 日	L2/L3 データ読込みを追記
第三版	平成 28 年 3 月 25 日	SST の物理量 2 層化に対応
第四版	平成 28 年 11 月 17 日	L1B/L1R Pixel Data Quality の読込方法変更に対応

目次

１．関連文書およびツールとデータの入手方法

本書では GCOM-W1/AMSR2 データの利用方法および必要な基礎知識について取扱いますが、GCOM-W1 衛星および AMSR2 プロダクトについてのより詳しい情報は、GCOM-W1 データ提供サービスホームページから以下の関連文書をダウンロードできます。

- GCOM-W1 データ利用ハンドブック
- AMSR2 レベル 1 処理プロダクトフォーマット説明書
- AMSE2 高次処理プロダクトフォーマット説明書(L3 プロダクト)

GCOM-W1 データ提供サービス
http://gcom-w1.jaxa.jp/

GCOM-W1 データ提供サービスホームページでは、AMSR2 プロダクトおよび過去の同一系統センサである AMSR/AMSR-E プロダクトを無償提供しています。ご利用にはユーザ登録が必要です。
本書で解説する AMSR2 Product I/O Toolkit (AMTK)、取扱い説明書、AMTK 用うるう秒ファイルについても GCOM-W1 データ提供サービスホームページからダウンロードできます。その他、GCOM-W1 関連情報については、GCOM ホームページをご覧ください。

GCOM ホームページ
http://suzaku.eorc.jaxa.jp/GCOM/

本書では、AMSR2 プロダクトのデータフォーマットとして採用されている HDF5 ライブラリについても取扱います。HDF5 ライブラリおよび関連する SZIP ライブラリについては、下記ホームページからダウンロードしてください。

The HDF Group ホームページ
http://www.hdfgroup.org/

本書で使用するサンプルプログラムは、下記 web ページからダウンロードできます。
http://suzaku.eorc.jaxa.jp/GCOM_W/data/data_w_use_j.html

２．本書の構成

本書ではGCOM-W1/AMSR2データを利用するために必要となるGCOM-W1衛星およびAMSR2センサの基礎知識について説明を行った後、ライブラリのインストール方法とサンプルプログラムの解説を行います。プログラム言語としてはC言語とFORTRAN90について取扱います。

AMSR2プロダクトはデータフォーマットしてHDF5形式を採用しており、HDF5ライブラリ経由でデータファイルにアクセスします。HDF5ライブラリ単体でもデータ利用は可能ですが、HDF5ライブラリをベースとした専用のツールキット(AMTK)も用意されています。

本書では、AMTKを利用する方法とHDF5ライブラリから直接データアクセスする方法を、C言語とFORTRAN90で解説します。サンプルプログラム解説は以下の4パターンで同じ内容を取り扱いますので、ご使用になる環境に合わせてご利用ください。

- AMTK編(C言語)
- AMTK編(FORTRAN90)
- HDF5編(C言語)
- HDF5編(FORTRAN90)

AMTKには、内部表現時刻形式の自動変換やAMSR2センサの各周波数毎の観測点緯度経度の算出機能、整数型格納変数の自動スケール変換機能が実装されています。時刻形式変換と各周波数緯度経度の算出については、本書のサンプルプログラムで同等のサブルーチンを用意していますので、HDF5ライブラリからも利用可能です。

３．GCOM-W1/AMSR2 データの基礎知識

３．１ 衛星軌道

GCOM-W1 衛星の軌道は、先行機である AMSR-E センサーが搭載された Aqua 衛星の軌道と同一です。GCOM-W1 衛星の軌道諸元を表 1 に、軌道イメージを図 1 に示します。

表 1　GCOM-W1 衛星軌道諸元

項目	仕様
軌道種別	太陽同期準回帰軌道
軌道高度	699.6km(赤道上)
軌道傾斜角	98.186 度
昇交点通過地方太陽時	13 時 30 分±15 分
回帰日数(周回数)	16 日(233 周)

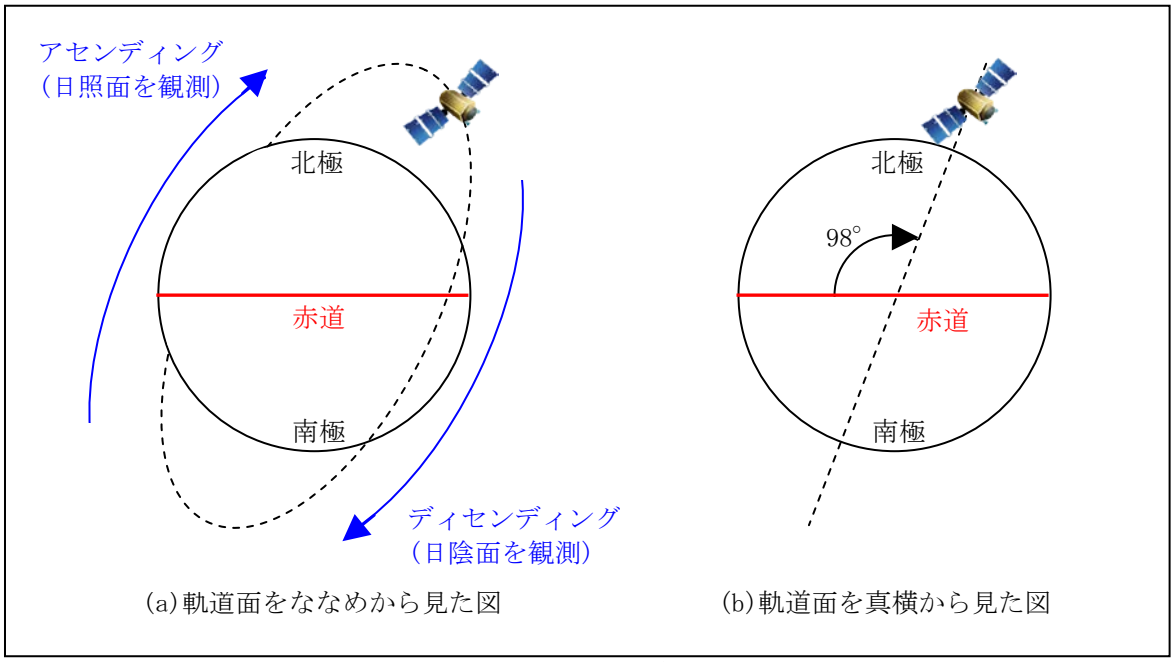
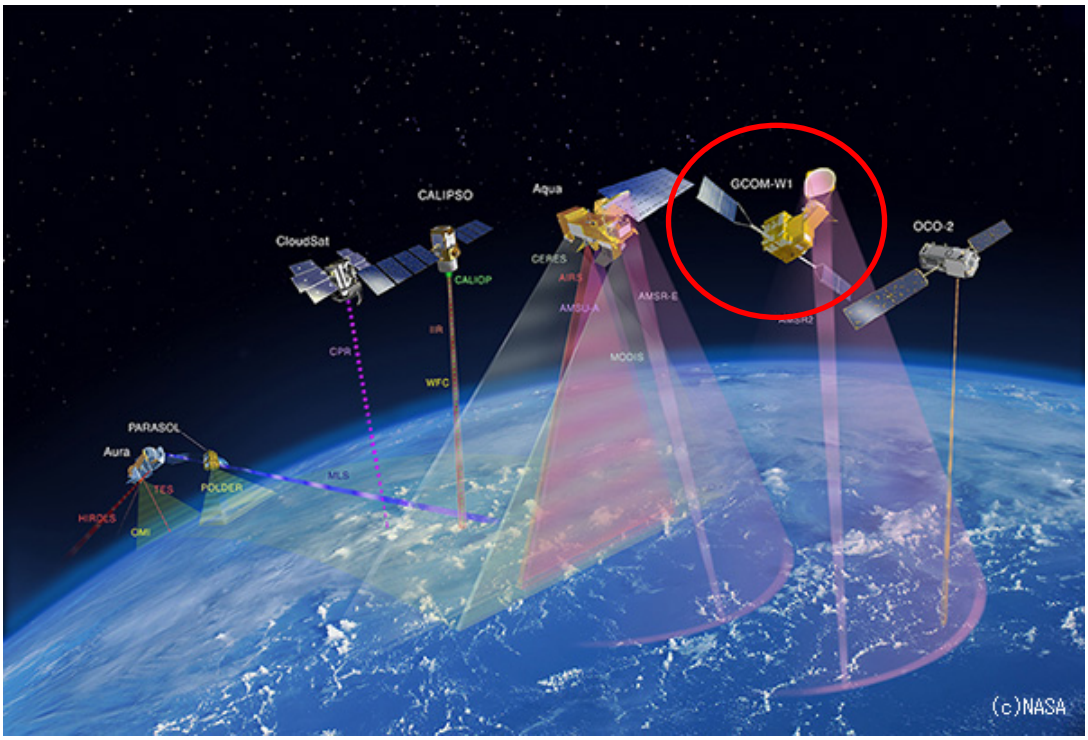


図 1　GCOM-W1 衛星軌道イメージ

GCOM-W1 衛星は赤道に対して約 98 度傾いた南北方向の軌道で地球を周回しています。
南極から北極への上昇をアセンディング、北極から南極への下降をディセンディングと呼びます。

GCOM-W1 衛星の軌道は、太陽同期準回帰軌道です。
太陽同期軌道とは、周期が太陽と同期している軌道です。アセンディングでは地球の日照面(昼)を、ディセンディングでは地球の日陰面(夜)を観測しており、同じ場所は同じ時刻に観測されます。アセンディングでの赤道交差時刻は、観測領域の地方時刻で 13 時 30 分頃となります。
準回帰軌道とは、一定の期間で同じ地表面上空に戻ってくる軌道です。GCOM-W1 衛星は、16 日間で地球を 233 周すると同じ地表面上空に戻ってきます。1 日では地球を約 15 周します。

◇参考：Ａ－Ｔｒａｉｎ



（A-Train とは）
A-Train（A-Train：The Afternoon Constellation）とは、高度約 700km、昇交点通過地方平均太陽時 13 時 30 分付近を観測軌道とする複数衛星から構成される NASA 主導の地球観測衛星のコンステレーション（衛星群）です。各国の衛星が協力して地球全体を観測するシステムで、現在軌道上で A-Train に参加している衛星は、Aqua（米 NASA）、CloudSat（米 NASA）、CALIPSO（米 NASA/仏 CNES）、Aura（米 NASA）であり、日本から初めて GCOM-W1（しずく）が参加します。

（A-Train の特長）
地球観測では、いろいろなセンサで同時に同地点を観測することが非常に効果的です。複数の衛星がほぼ同一軌道に隊列を成して飛行することにより、複数の衛星で地球の同一地点をほぼ同じ時刻（約 10 分以内）に観測することができます。
各衛星の位置は厳密に管理されており、A-Train 軌道に入るには、すでに入っている人工衛星を避けて、決められた位置に入る必要があります。GCOM-W1 は、JAXA が培ったランデブーの技術を活かし、A-Train 軌道に入りました。人工衛星をこのようなコンステレーションに投入し、運用するのは、JAXA として初めてのことです。

≪A-Train（衛星コンステレーション）参加衛星≫
●Aura　NASA（米国）　2004 年 7 月 15 日　打上げ
　地球大気の組成、化学反応、ダイナミクスを解明するための観測データを取得する衛星
●CALIPSO　NASA/CNES　（米国／フランス）2006 年 4 月 28 日　打上げ
　エアロゾルや雲が地球の気候に与える影響を解明するための観測データを取得する光学ライダー衛星
●CloudSat　NASA　（米国）2006 年 4 月 28 日　打上げ
　雲が地球の気候に与える影響を解明するための観測データを取得する電波レーダ衛星
●Aqua　NASA　（米国）2002 年 5 月 4 日　打上げ
　ラテン語で水（アクア）を意味する。大気中及び海からの水蒸気、雲、降雨、海氷、土壌水分等の
　様々な地球の水循環に関する観測データを取得する衛星
●OCO-2　NASA（米国）大気中の二酸化炭素（CO2）量を観測（予定）

３．２ 観測パス(グラニュール)

GCOM-W1/AMSR2 標準プロダクトデータは、観測緯度に対する極から極までの半周回を 1 つの観測パス(グラニュール)として作成されます。大きく分けて、北極から南極への半周回(ディセンディング)と南極から北極への半周回(アセンディング)の 2 種類があります。ディセンディングは日陰面(夜)の観測で、アセンディングは日照面(昼)の観測になります。

図 2 に、回帰 1 日目の標準プロダクトの観測パス領域を示します。全球マップが描かれている部分が観測領域で、白い隙間は観測外領域です。

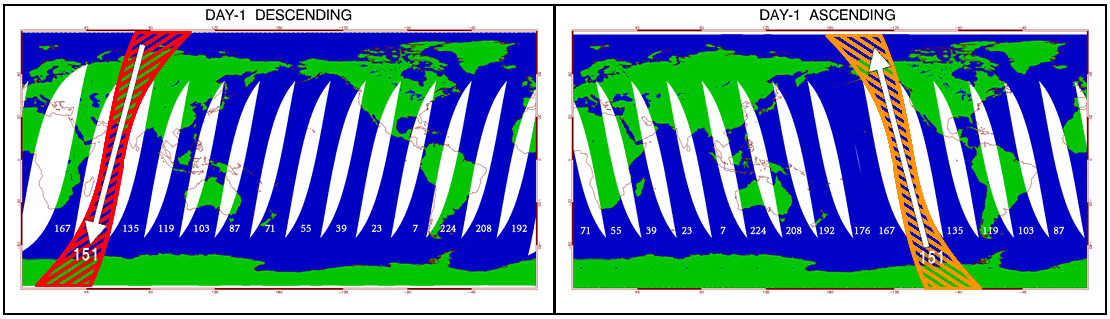


図 2　標準プロダクトの観測パス領域

観測領域に表示されている数字はパス番号です。GCOM-W1 衛星が回帰日数 16 日の間に地球を周回する 233 周には、1～233 の固有番号が付けられており、これをパス番号と呼びます。連続する周回では、パス番号は 16 ずつ増えていきます。標準プロダクトの観測パスは半周回のパスなので、ディセンディングの D やアセンディングの A を付加して「151D」や「151A」のように呼ばれます。図 2 の赤色の部分は「151D」の観測領域、橙色の部分は「151A」の観測領域です。観測パスおよびパス番号の定義は、先行機である Aqua/AMSR-E と同一です。

◇参考：観測パス開始位置とパス番号開始位置

観測パスの開始位置は極ですが、パス番号の開始位置はアセンディング赤道交差点として定義されています。観測パスに付与されるパス番号は、観測パス開始位置でのパス番号となります。

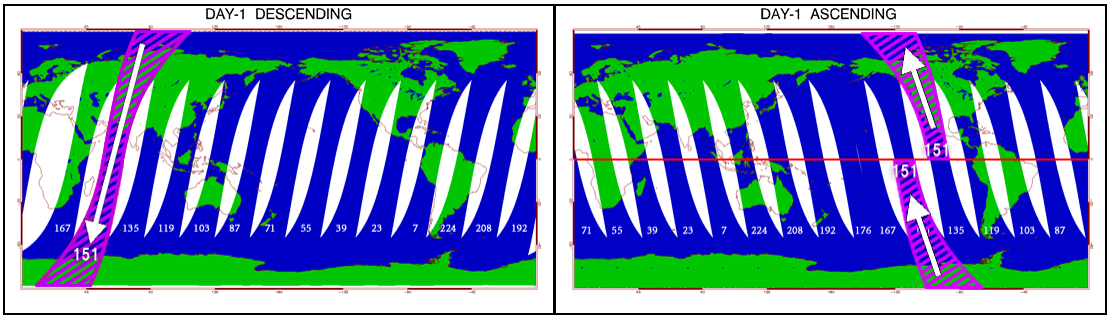


図 3　パス番号の開始位置

例として、パス番号 151 の領域を図 3 に紫色で示します。
観測パス「151D」は観測開始位置(北極)において、パス番号が 151 の領域から始まっています。
観測パス「151A」は観測開始位置(南極)において、パス番号が 151 の領域から始まっています。

３．３ L1 データと高次データ

AMSR2 センサーはマイクロ波放射計です。観測するのは 6.9GHz～89.0GHz の電磁波で、観測されたデータは、観測データカウント値(L1A)→輝度温度(L1B)→地球物理量(L2)と変換されます。AMSR2 プロダクトで算出される地球物理量は表 2 の通りです。

表 2　AMSR2 プロダクトで算出される地球物理量

物理量名
可降水量　(Total Precipitable Water)
積算雲水量(Cloud Liquid Water)
降水量　(Precipitation)
海面水温　(Sea Surface Temperature)
海上風速　(Sea Surface Wind Speed)
海氷密接度(Sea Ice Concentration)
積雪深　(Snow Depth)
土壌水分量(Soil Moisture Content)

L1 は輝度温度(Brightness Temperature; TB)の観測パスデータです。
L2 は地球物理量(Geophysical Data; GEO)に変換された観測パスデータです。
L3 は地図投影されたデータです。L3 には輝度温度(L3TB)と地球物理量(L3GEO)があります。L3 には日集計データと月集計データがあり、アセンディングとディセンディングに分けて作成されます。地球物理量として算出されるプロダクト(L2 と L3GEO)は高次データと呼ばれます。

L1/L2 データは、観測パスの矩形データです。この矩形データを、観測点の緯度経度を元に地図投影して L3 データが作成されます。図 4 に地図投影前後の比較図を示します。

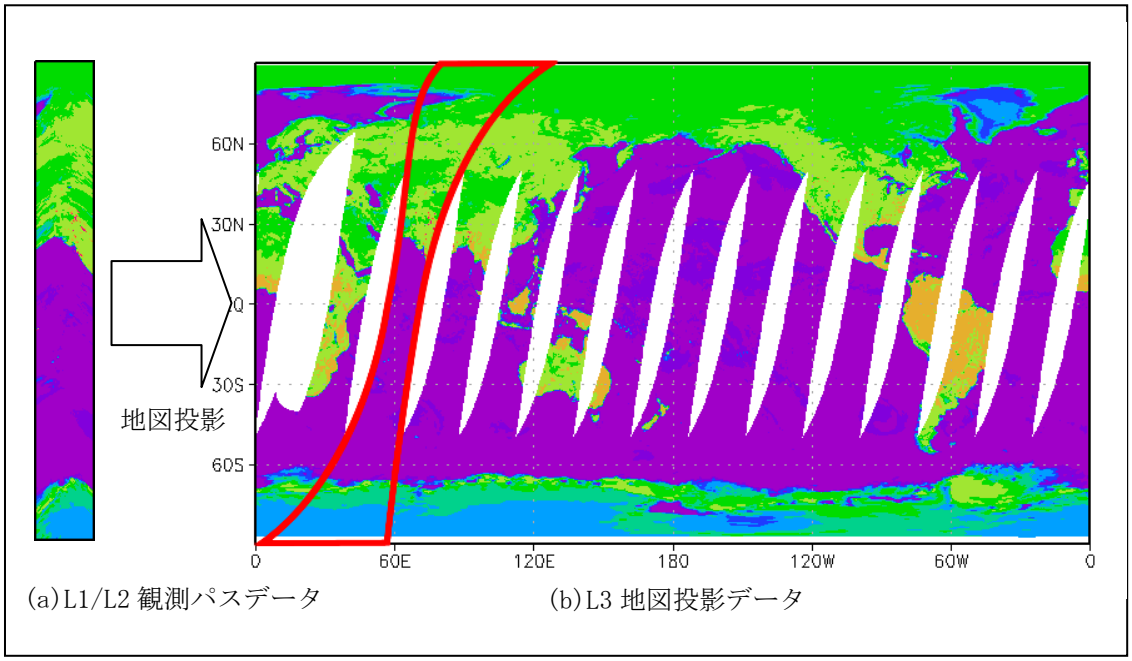


図 4　観測パスの地図投影

３．４ ファイル名規則

AMSR2 プロダクトのファイル名は、グラニュール ID＋拡張子".h5"となっています。
L1/L2 グラニュール ID を図 5 に、L3 グラニュール ID を図 6 に示します。

```
                          1         2         3         4
文字位置:     12345678901234567890123456789012345678901
ファイル名: GW1AM2_YYYYMMDDHHmm_PPPX_LLxxKKKrdvaaappp
L1 サンプル: GW1AM2_201209071216_068D_L1SGBTBR_0000000
L2 サンプル: GW1AM2_201209071216_068D_L2SGSSTLA0000000

GW1 : 衛星名(GW1 固定)  AM2 : センサ種別(AM2 固定)  YYYYMMDDHHmm : 観測開始日時(年は西暦、時刻は UT)
PPP : パス番号(001～233、開始時のパス番号)  X : 軌道(A:アセンディング、D:ディセンディング、B:両方)
LL  : 処理レベル(L1:レベル 1、L2:レベル 2)  xx : 処理種別(SG:標準、SN:準リアル、SL:準リアル日本周辺)
KKK : プロダクト ID
      L1 : ADN -> L1A、BTB -> L1B、RTB -> L1R
      L2 : CLW -> 積算雲水量(Cloud Liquid Water)      PRC -> 降水量(Precipitation)
           SIC -> 海氷密接度(Sea Ice Concentration)   SMC -> 土壌水分量(Soil Moisture Content)
           SND -> 積雪深(Snow Depth)                  SST -> 海面水温(Sea Surface Temperature)
           SSW -> 海上風速(Sea Surface Wind Speed)    TPW -> 可降水量(Total Precipitable Water)
r   : 解像度(L1 では R[Raw]固定、L2 では L[低解像度 243 点]または H[高解像度 486 点])
d   : 開発者 ID(L1 では_、L2 では A～Z)   v   : プロダクトバージョン(0～9、A～Z)
aaa : アルゴリズムバージョン(000～999)  ppp : パラメータバージョン(000～999)
```

図 5　L1/L2 グラニュール ID

```
                          1         2         3         4
文字位置:                 12345678901234567890123456789012345678901
ファイル名:               GW1AM2_YYYYMMDD_ttt_PPWX_LLxxKKKrdvaaappp
L3 サンプル(物理量)  : GW1AM2_20120907_01D_EQ0A_L3SGSSTLA0000000
L3 サンプル(輝度温度): GW1AM2_20120907_01D_EQMA_L3SGT06LA0000000

GW1 : 衛星名(GW1 固定)  AM2 : センサ種別(AM2 固定)  YYYYMMDD : 観測開始日(年は西暦、時刻は UT)
ttt : 統計期間(01D:日単位、01M:月単位[DD は 00])
PP  : 地図投影種別(EQ:等緯度経度、PN:ポーラステレオ北半球、PS:ポーラステレオ南半球)
W   : 統計手法(M:平均、0:最新上書き)  X : 軌道(A:アセンディング、D:ディセンディング)
LL  : 処理レベル(L3:レベル 3)  xx : 処理種別(SG:標準)
KKK : プロダクト ID
      T06,T07,T10,T18,T23,T36,T89 -> 各周波数輝度温度(1 ファイルに H と V が格納されます)
      CLW -> 積算雲水量(Cloud Liquid Water)      PRC -> 降水量(Precipitation)
      SIC -> 海氷密接度(Sea Ice Concentration)   SMC -> 土壌水分量(Soil Moisture Content)
      SND -> 積雪深(Snow Depth)                  SST -> 海面水温(Sea Surface Temperature)
      SSW -> 海上風速(Sea Surface Wind Speed)    TPW -> 可降水量(Total Precipitable Water)
r   : 解像度(L:低解像度、H:高解像度)
```

地図投影種別	低解像度ピクセル数		高解像度ピクセル数	
	経度または横	緯度または縦	経度または横	緯度または縦
EQ	1440	720	3600	1800
PN(輝度温度、SIC)	304	448	760	1120
PS(輝度温度、SIC)	316	332	790	830
PN(SND)	432	574	1080	1435

```
d   : 開発者 ID(A～Z)   v   : プロダクトバージョン(0～9、A～Z)
aaa : アルゴリズムバージョン(000～999)  ppp : パラメータバージョン(000～999)
```

図 6　L3 グラニュール ID

３．５ 観測スキャンとオーバーラップ

観測スキャン概要図を図 7 に示します。AMSR2 センサーは、衛星進行方向に対して、反時計回りの円弧上に地表面をスキャンしていきます。1 スキャンには複数の観測点があります。L1/L2 データには、連続する観測スキャンが二次元配列データとして格納されています。

オーバーラップ概要図を図 8 に示します。L1/L2 は観測緯度に対する極から極までの半周回データですが、L1 では極の前後 20 スキャンがオーバーラップとして重複して格納されています。L2 には重複部分はありません。同じパスの L1 と L2 を比較すると、L1 の方が 40 スキャン多くデータが格納されています。

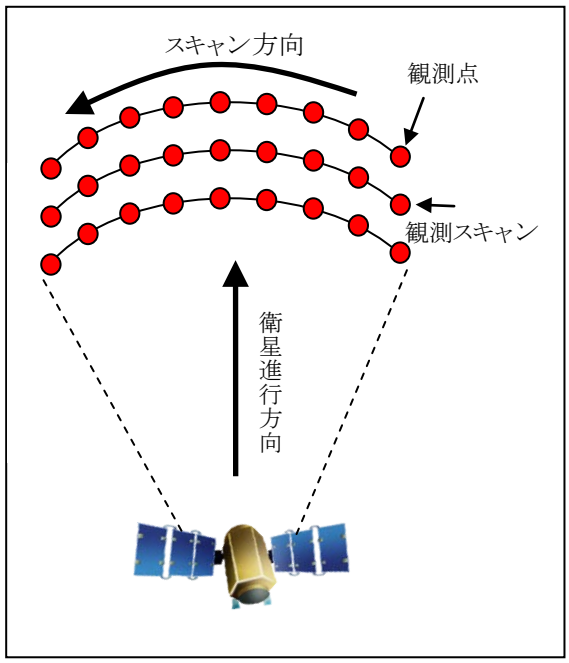


図 7　観測スキャン概要図

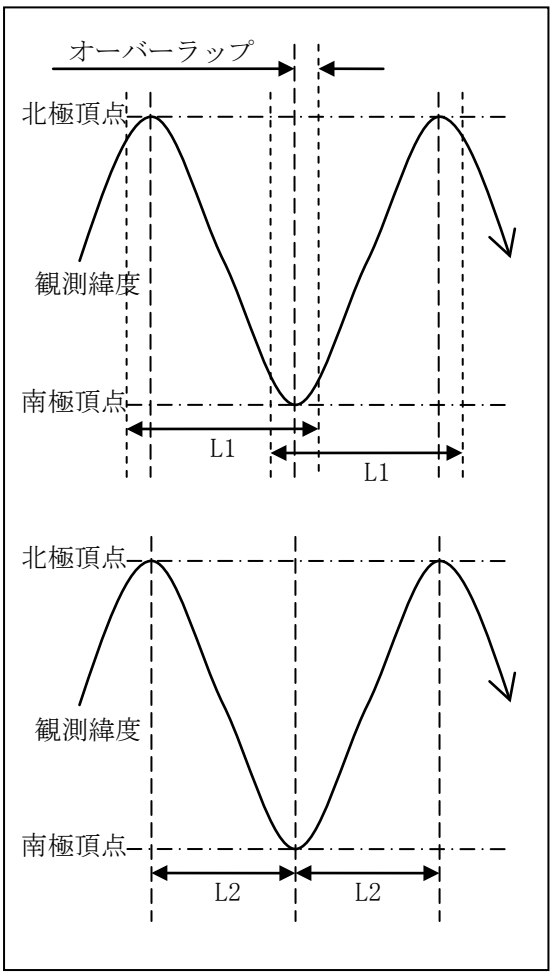


図 8　オーバーラップ概要図

３．６ 時刻形式

AMSR2 プロダクトの L1/L2 には、観測スキャン毎の時刻が TAI93 形式で格納されています。TAI93 形式とは、うるう秒も含めた、1993 年 1 月 1 日 0 時 0 分(UT)からの通算秒です。

うるう秒とは、地球自転の揺らぎにより発生する世界時刻と原子時計時刻との差異を調整するために追加または削除される秒です。うるう秒は地球自転の揺らぎにより不規則に発生しますので、うるう秒が追加または削除された日付の一覧(うるう秒ファイル)がなければ TAI93 形式の変換はできません。

例えば 2012 年 7 月 24 日は、1993 年 1 月 1 日からの通算秒が 617,241,600 で、この間に 8 回うるう秒が追加されていますので、TAI93 形式では 617,241,608 となります。

AMTK を使用する場合は、時刻データは年/月/日/時/分/秒形式に自動的に変換されますが、HDF5 ライブラリを使用して格納データを直接読込む場合は、時刻変換処理を別途行う必要があります。(本文書の HDF5 編サンプルプログラムには TAI93 時刻変換サブルーチンが付属しています。)

３．７ 観測周波数とフットプリント

AMSR2 センサーが観測する周波数帯は、6.9GHz、7.3GHz、10.7GHz、18.7GHz、23.8GHz、36.5GHz、89.0GHz の 7 つあります。各周波数帯には、水平偏波チャンネルと、垂直偏波チャンネルがあります。さらに、89.0GHz では A ホーンと B ホーンに分かれて観測されます。合計で観測チャンネルは 16 になります。表 3 に観測チャンネル一覧を示します。

表 3　AMSR2 センサーの観測チャンネル

1	6.9GHz 水平偏波	低周波観測 1 スキャンは 243 点
2	6.9GHz 垂直偏波	
3	7.3GHz 水平偏波	
4	7.3GHz 垂直偏波	
5	10.7GHz 水平偏波	
6	10.7GHz 垂直偏波	
7	18.7GHz 水平偏波	
8	18.7GHz 垂直偏波	
9	23.8GHz 水平偏波	
10	23.8GHz 垂直偏波	
11	36.5GHz 水平偏波	
12	36.5GHz 垂直偏波	
13	89.0GHz A ホーン 水平偏波	高周波観測 1 スキャンは 486 点
14	89.0GHz A ホーン 垂直偏波	
15	89.0GHz B ホーン 水平偏波	
16	89.0GHz B ホーン 垂直偏波	

6.9GHz～36.5GHz は低周波観測で、1 スキャンは 243 点です。89.0GHz は高周波観測で、1 スキャンは 486 点です。物理量算出で参照される観測チャンネルは物理量によって異なりますが、主となる観測チャンネルによって、L2 サンプル数が異なります。表 4 に各物理量の L2 サンプル数を示します。

表 4　各物理量の L2 サンプル数

物理量名	L2 サンプル数
可降水量　(Total Precipitable Water)	243 点
積算雲水量(Cloud Liquid Water)	243 点
降水量　(Precipitation)	486 点
海面水温　(Sea Surface Temperature)	243 点
海上風速　(Sea Surface Wind Speed)	243 点
海氷密接度(Sea Ice Concentration)	243 点
積雪深　(Snow Depth)	243 点
土壌水分量(Soil Moisture Content)	243 点

1 つの観測点で観測される地表面領域はフットプリントと呼ばれますが、フットプリントの大きさは周波数によって異なります。表 5 と図 9 に各周波数のフットプリントを示します。

表 5　各周波数のフットプリント

周波数	フットプリント (スキャン方向×衛星進行方向)
6.9GHz	35km × 62km
7.3GHz	34km × 58km
10.7GHz	24km × 42km
18.7GHz	14km × 22km
23.8GHz	15km × 26km
36.5GHz	7km × 12km
89.0GHz	3km × 5km

注）18.7GHz のフットプリントは、23.8GHz より小さくなります。

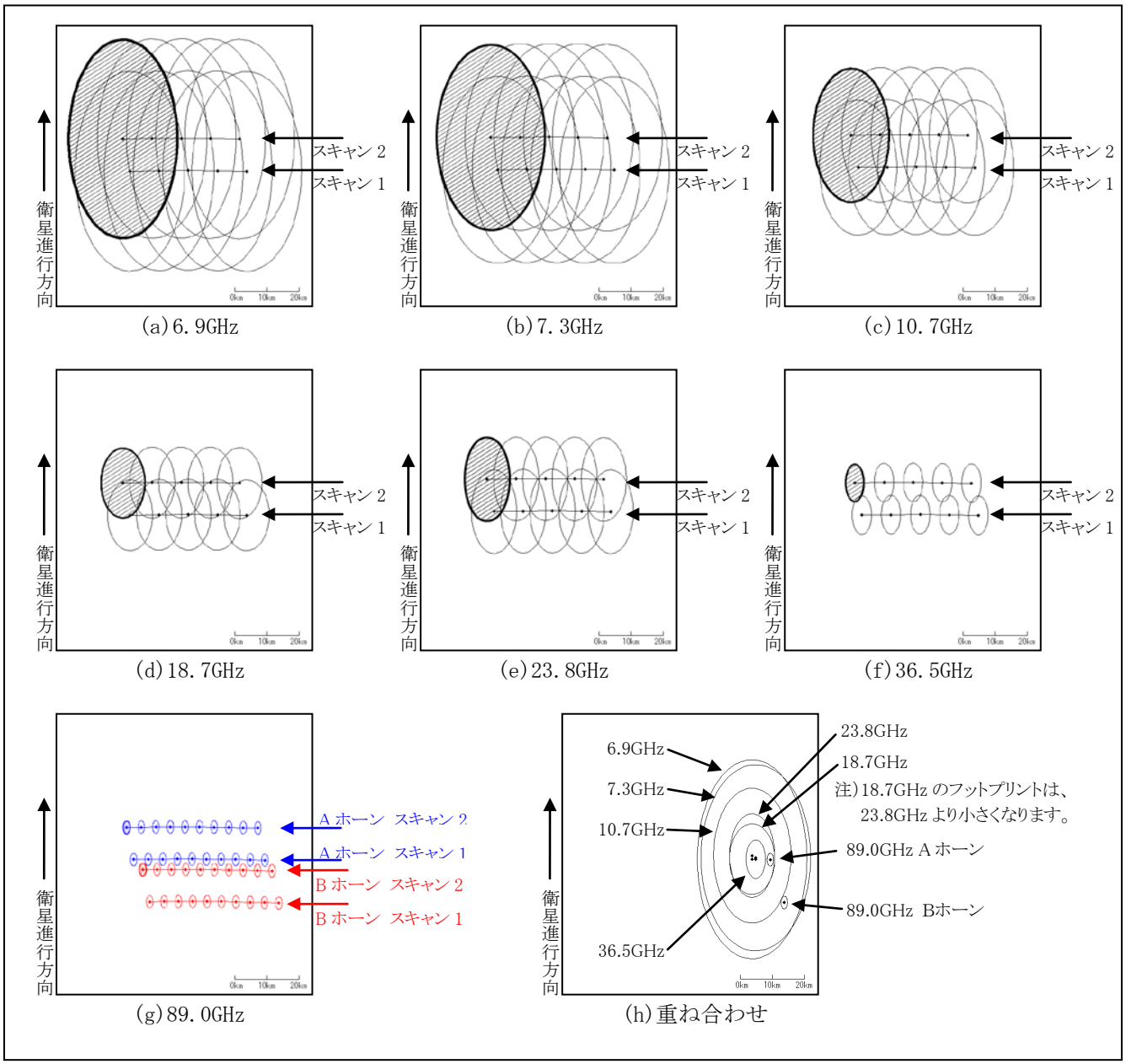


図 9　各周波数のフットプリントサンプル

同じ観測点でも、厳密には周波数毎に中心緯度経度が異なります。
図 10 に、同じ観測点の中心緯度経度の比較サンプルを示します。

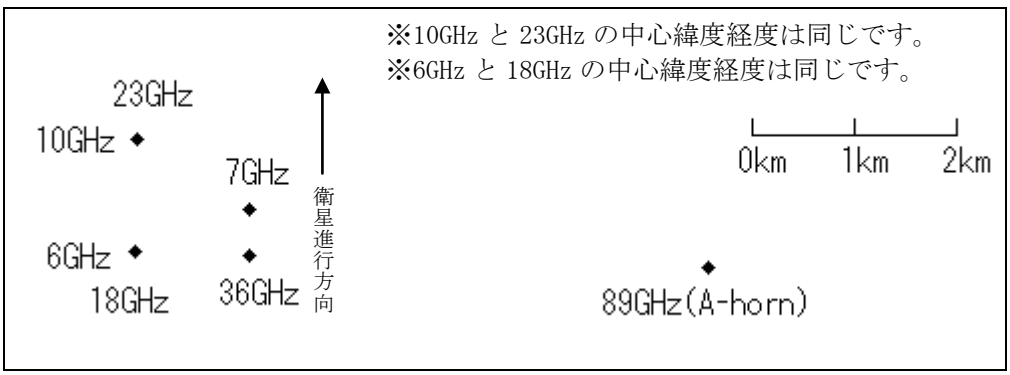


図 10　同じ観測点の中心緯度経度の比較サンプル

３．８ L1 リサンプリングデータ(L1R)

物理量算出等の観測データ処理では複数の周波数データを組合せて使用しますが、その場合、異なるフットプリントのデータを組合せるので、厳密には対象となる地表面領域の整合が取れていません。
AMSR2 プロダクトでは、対象となる地表面領域の整合が取れたデータを提供するため、特定の周波数のフットプリントを基準にして、各周波数のフットプリントをリサンプリングしたデータ(L1R)を用意しています。
L1R では、観測点の中心緯度経度も統一しており、89G A ホーン緯度経度を基準としています。
基準となるフットプリントは 6G/10G/23G/36G の 4 種類で、基準フットプリントより小さいフットプリント(基準周波数より高い周波数)がリサンプリングデータとして算出されます。表 6 に L1R 輝度温度データ一覧を示します。

表 6　L1R 輝度温度データ一覧

フットプリント	6G (H/V) 243 点	7G (H/V) 243 点	10G (H/V) 243 点	18G (H/V) 243 点	23G (H/V) 243 点	36G (H/V) 243 点	89G (H/V) 243 点	89G A-horn (H/V) 486 点	89G B-horn (H/V) 486 点
6G 解像度	☆	○	○	○	○	○	○		
10G 解像度			☆	○	○	○	○		
23G 解像度				○	☆	○	○		
36G 解像度						☆	○		
89G 解像度								△	△

○は中心緯度経度とフットプリントをリサンプリングしたデータ
☆は中心緯度経度をリサンプリングしたデータ
△は観測オリジナルデータ
注）18G は 23G よりフットプリントが小さいため、23G 解像度に含まれます。

L1R には、6G/10G/23G/36G 解像度のリサンプリングデータに加えて、89G 解像度としてオリジナルの 89G データも含まれます。
観測点の中心緯度経度を統一してリサンプリングしているため、6G 解像度の 6G データでも、オリジナルの 6G データとは一致しません。

３．９ 陸海フラグ

AMSR2 プロダクトには、補助データとして観測点の陸海情報が格納されています。この陸海情報は陸海フラグと呼ばれており、フットプリントに対する陸の割合が 0%～100%の整数で格納されています。水域は 0%、陸域は 100%です。図 11 に陸海フラグ概要図を示します。

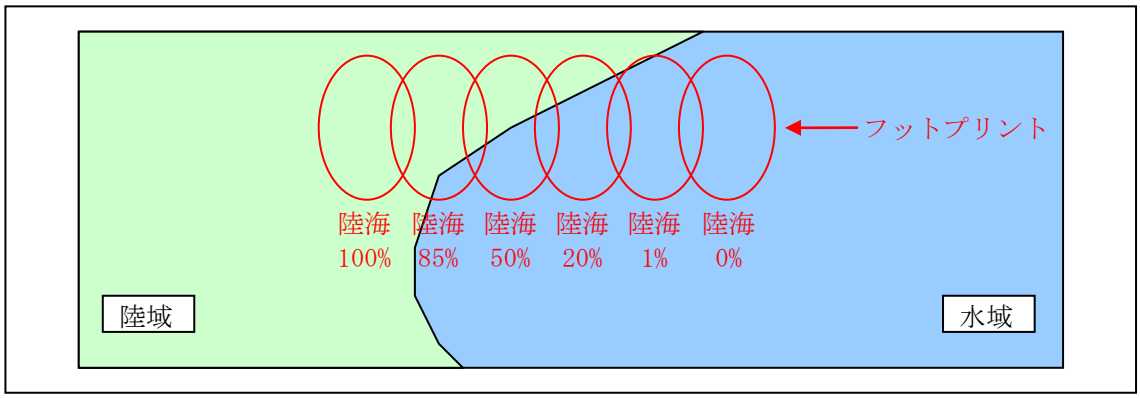


図 11　陸海フラグ概要図

高周波用陸海フラグは、A ホーン用と B ホーン用の 2 種類あります。
低周波用陸海フラグは、L1B では 6 種類の周波数(フットプリント)があるので、陸海フラグも 6 種類(6G/7G/10G/18G/23G/36G)あります。L1R では 4 種類の解像度(フットプリント)があるので、陸海フラグも 4 種類(6G/10G/23G/36G)あります。

３．１０ L1/L2 に格納されている緯度経度

L1 に格納されている緯度経度は、89G A ホーン緯度経度と 89G B ホーン緯度経度です。
L1B では、緯度経度は周波数毎に異なりますが、低周波の各周波数の緯度経度は、89G A ホーン緯度経度と相対レジストレーション係数を用いて算出できます。
L1R は 89G A ホーン緯度経度を基準にリサンプリングされているので、低周波の各解像度の緯度経度は全て同じで 89G A ホーン緯度経度を使用します。ただし、低周波観測が 243 点なのに対し、高周波観測は 486 点なので、1 から数えて奇数番目の緯度経度を使用します。
L1B の低周波の緯度経度は、AMTK を使用する場合は自動的に算出されますが、HDF5 ライブラリを使用する場合は緯度経度計算を別途行う必要があります。(本文書の HDF5 編サンプルプログラムには低周波緯度経度算出サブルーチンが付属しています。)
L1R の低周波用緯度経度は、AMTK を使用する場合は自動的に取得されますが、HDF5 ライブラリを使用する場合は 89G A ホーン緯度経度の間引き処理を別途行う必要があります。

L2 に格納されている緯度経度は、L1B を入力とする物理量と、L1R を入力とする物理量で異なります。(入力となった L1 は、L2 プロダクトメタデータの InputFileName で確認できます。)
L1B を入力とする物理量では、輝度温度の緯度経度は周波数毎に異なるので、物理量の緯度経度としては低周波緯度経度の平均値が格納されています。
L1R を入力とする物理量では、輝度温度の緯度経度は全ての解像度で統一されているので、物理量の緯度経度としては L1R の低周波用緯度経度が格納されています。
L2 降水量プロダクトは高解像度 L2 プロダクトなので、89G A ホーン緯度経度と 89G B ホーン緯度経度が格納されています。

L1/L2 に格納されている緯度経度について表 7 にまとめます。

表 7　L1/L2 に格納されている緯度経度

プロダクト	格納されている緯度経度	備考
L1B	89G A ホーン緯度経度 89G B ホーン緯度経度	低周波の各周波数の緯度経度は、相対レジストレーション係数を用いて算出します。
L1R	89G A ホーン緯度経度 89G B ホーン緯度経度	低周波の緯度経度は、89G A ホーン緯度経度の 1 から数えて奇数番目を使用します。
L2(L1B 入力)	L1B 低周波平均緯度経度	
L2(L1R 入力)	L1R 低周波用緯度経度 (89G A ホーンの奇数番目)	
L2(降水量)	89G A ホーン緯度経度 89G B ホーン緯度経度	

３．１１ 緯度経度の標高補正(L1R)

L1B では、観測点の緯度経度を地球楕円体とセンサー視線の交点として求めており、地表面高さは考慮していません。AMSR2 の視線は地表面に対して斜めに入射するので、標高が高い地域では実際の観測点とのずれが大きくなります。
L1R では、L1B 緯度経度に対して標高補正を行って、地表面高さによる緯度経度のずれを補正しています。地表面高さをH、観測入射角を θ とした時、水平移動距離Lを次の式で計算して補正を行います。

$$L = H\tan\theta$$

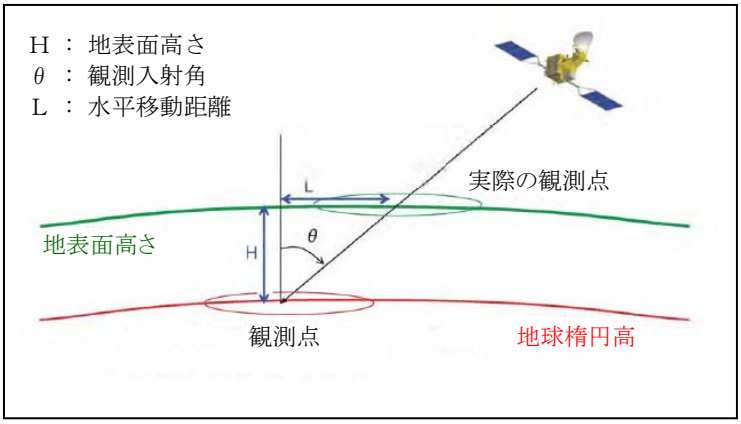


図 12　標高補正

３．１２ 主なデータの HDF5 格納型・スケール・欠損値

表 8 に主なデータの HDF5 格納型・スケール・欠損値を示します。

表 8　主なデータの型・スケール・欠損値

データ	HDF5 格納型	スケール[単位]	欠損値	L3 観測外領域
L1/L2 時刻(TAI93)	8byte 浮動小数点型	1[sec]	-9999.0	
L1/L2 緯度経度	4byte 浮動小数点型	1[deg]	-9999.0	
L1 観測入射角	2byte 整数型(符号あり)	0.01[deg]	-32768	
L1 観測方位角	2byte 整数型(符号あり)	0.01[deg]	-32768	
L1/L3 輝度温度	2byte 整数型(符号なし)	0.01[K]	65535	65534
L2/L3 物理量	2byte 整数型(符号あり)	可降水量: 0.01[kg/m2] 積算雲水量: 0.001[kg/m2] 降水量: 0.1[mm/h] 海面水温: 0.01[degC] 海上風速: 0.01[m/s] 海氷密接度: 0.1[%] 積雪深: 0.1[cm] 土壌水分量: 0.1[%]	-32768	-32767

データファイルのサイズが小さくなるように、輝度温度や物理量の格納型には 2byte 整数型が使われています。2byte 整数型の値の範囲は、符号なしの場合は 0～65535、符号ありの場合は-32768～32767 となるので、その範囲で目的のデータを表現するために、スケールが設定されています。

例えば、輝度温度として 28312 が格納されていた場合、輝度温度のスケールは 0.01[K]なので、実際の輝度温度は 283.12[K]となります。

AMTK を使用する場合はスケールは自動的に処理されますが、HDF5 ライブラリを使用する場合はスケール処理を別途行う必要があります。

なんらかの理由によりデータが取得されなかった場合には欠損値が格納されます。例えば物理量の場合は、算出対象となる領域が限定されている場合があり、対象外領域には欠損値が格納されます。積算雲水量の場合、算出対象となるのは水域だけなので、陸域は対象外領域として欠損値-32768 が格納されます。

L1/L2 を地図投影して L3 を作成する場合は、欠損値と観測されていない領域を区別するため、観測外領域に欠損値と異なる値が設定されます。図 13 に欠損値と L3 観測外領域のサンプルを示します。

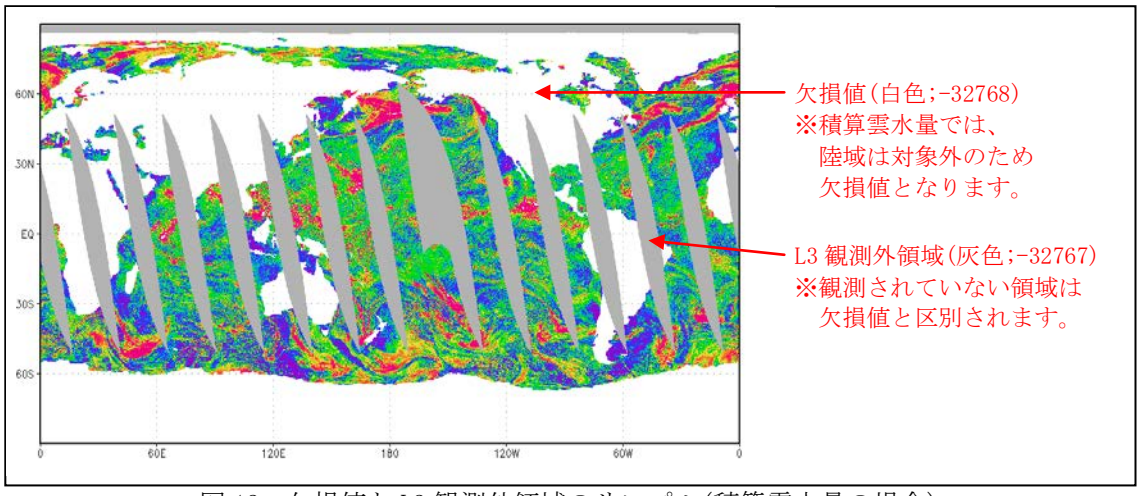


図 13　欠損値と L3 観測外領域のサンプル(積算雲水量の場合)

４．ライブラリのインストール

本書のサンプルプログラムは以下の環境で動作確認を行っています。

表 9　動作確認環境

計算機	Intel(R) Xeon(R) CPU E5504
ＯＳ	Red Hat Enterprise Linux release 5.4
C コンパイラ	GNU C compiler 4.1.2 Intel C compiler Version 11.1 PGI C compiler Version 10.9
FORTRAN コンパイラ	Intel FORTRAN compiler Version 11.1 PGI FORTRAN compiler Version 10.9
HDF5 SZIP	HDF5 version 1.8.4-patch1 SZIP version 2.1
AMTK	version 1.11

以下の手順では、linux 計算機において、ユーザー権限のみでインストールを行う方法について説明します。インストール先は/home/user1/util 配下とし、ライブラリ毎にディレクトリを作成していきます。図 14 にライブラリディレクトリ構成を示します。

このようなディレクトリ構成でインストールを行っておくと、後日、ライブラリがバージョンアップされた場合に、旧バージョンと新バージョンが共存でき、ライブラリパスを切換える事により、新/旧両バージョンを使い分ける事ができます。

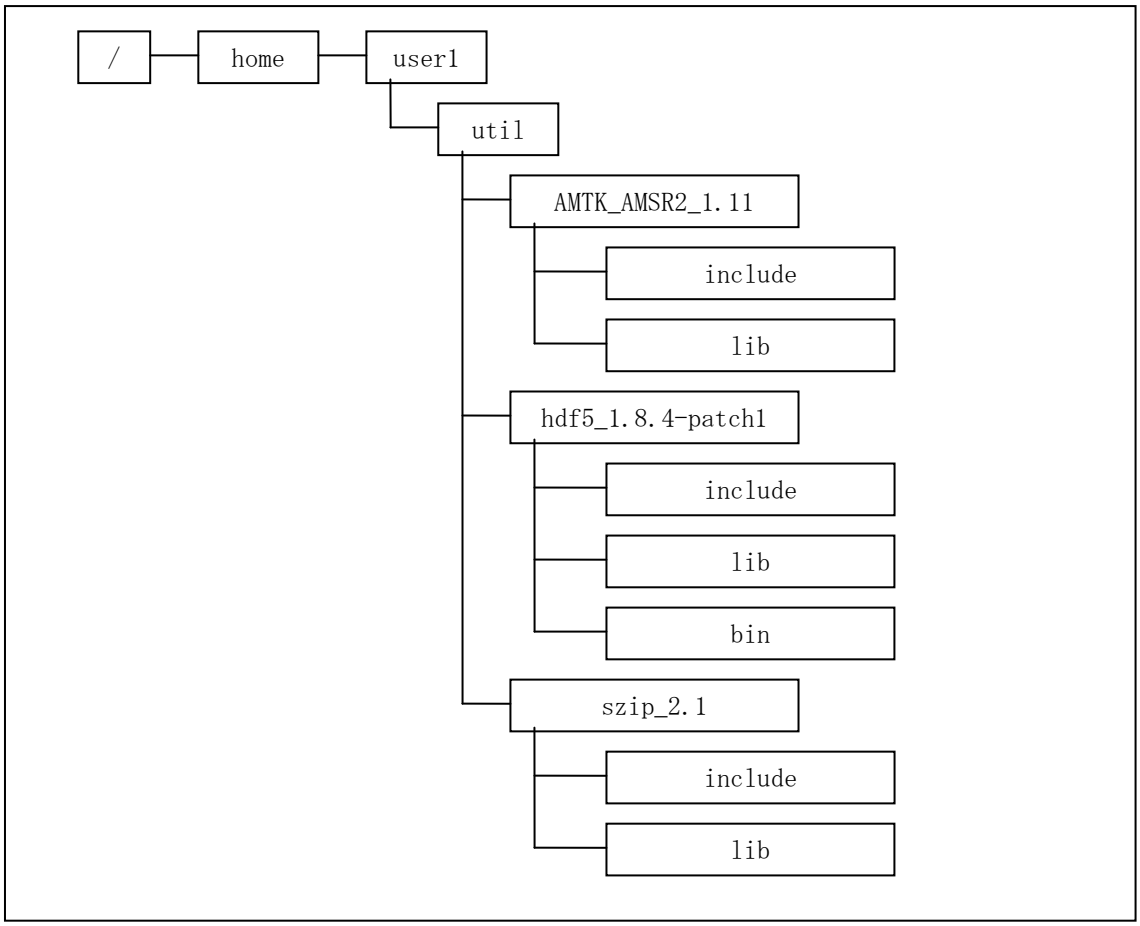


図 14　ライブラリディレクトリ構成

以下の説明では、/home/usr1 を自分のユーザーディレクトリに置き換えて作業してください。

(1)SZIP のインストール

1-1　ダウンロード
The HDF Group ホームページの SZIP ページ(http://www.hdfgroup.org/ftp/lib-external/szip/)からソースインストール版の圧縮ファイルをダウンロードします。
※以下では szip-2.1.tar.gz をダウンロードしたものとして説明します。

1-2　解凍
適当な作業ディレクトリで圧縮ファイルを解凍します。
以下のコマンドで解凍できます。
```
$ tar -xzvf szip-2.1.tar.gz
```
解凍すると、szip-2.1 のようなディレクトリが作成されるので、その配下へ移動します。
```
$ cd szip-2.1
```

1-3　コンパイルとインストール
以下のコマンドを順番に実行して、コンパイルとインストールを行います。
--prfix=には、インストール先ディレクトリを指定します。
※この例の場合、szip のバージョンは 2.1 なので、szip_2.1 としています。
　バージョン文字部分は実際に使用するバージョンに置き換えてください。
```
$ ./configure --disable-shared --prefix=/home/user1/util/szip_2.1
$ make
$ make install
```

(2)HDF5 のインストール

2-1　ダウンロード
The HDF Group ホームページ(http://www.hdfgroup.org/)から HDF5 のソースインストール版の圧縮ファイルをダウンロードします。
※以下では hdf5-1.8.4-patch1.tar.gz をダウンロードしたものとして説明します。

2-2　解凍
適当な作業ディレクトリで圧縮ファイルを解凍します。
以下のコマンドで解凍できます。
$ tar -xzvf hdf5-1.8.4-patch1.tar.gz
解凍すると、hdf5-1.8.4-patch1 のようなディレクトリが作成されるので、その配下へ移動します。
$ cd hdf5-1.8.4-patch1

2-3　コンパイルとインストール
以下のコマンドを順番に実行して、コンパイルとインストールを行います。
--prfix=には、インストール先ディレクトリを指定します。
※この例の場合、hdf5 のバージョンは 1.8.4-patch1 なので、hdf5_1.8.4-patch1 としています。
　バージョン文字部分は実際に使用するバージョンに置き換えてください。

＜HDF5 の FORTRAN ライブラリを使用しない場合＞
```
$ ./configure --disable-shared --prefix=/home/user1/util/hdf5_1.8.4-patch1
              --with-szlib=/home/user1/util/szip_2.1
$ make
$ make install
```

＜HDF5 の FORTRAN ライブラリを使用する場合＞
```
$ ./configure --disable-shared --prefix=/home/user1/util/hdf5_1.8.4-patch1
              --with-szlib=/home/user1/util/szip_2.1 --enable-fortran FC=ifort
$ make
$ make install
```

FORTRAN90 を使用する場合で、AMTK を使用せず、HDF5 ライブラリのみで AMSR2 プロダクトへアクセスしたい場合は、＜HDF5 の FORTRAN ライブラリを使用する場合＞の手順でインストールを行ってください。FC=には使用するフォートランコンパイラを指定します。ここではインテルコンパイラ(ifort)を指定しています。フォートランコンパイラは、インテルコンパイラ(ifort)または PGI コンパイラ(pgf90)で動作確認を行っています。

AMTK を使用しない場合は、(3)をスキップして、(4)を実施してください。
(3)を実施した場合は、(4)は必要ありません。

(3)AMTK のインストール

3-1　ダウンロード
GCOM-W1 データ提供サービスホームページ(http://gcom-w1.jaxa.jp/)から
インストール用圧縮ファイルをダウンロードします。
※以下では AMTK_AMSR2_Ver1.11.tar.gz をダウンロードしたものとして説明します。

注意事項)
SZIP/HDF5 では、圧縮ファイルを解凍した場所とインストール先は別ですが、
AMTK では、圧縮ファイルを解凍した場所がライブラリディレクトリになります。

3-2　解凍
/home/user1/util 配下に AMTK_AMSR2_Ver1.11.tar.gz を配置して、
以下のコマンドを実行します。
$ tar -xzvf AMTK_AMSR2_Ver1.11.tar.gz
解凍すると、AMTK_AMSR2 のようなディレクトリが作成されるので、
AMTK_AMSR2_1.11 のようなバージョン付きの名前にリネームします。
リネーム後、その配下へ移動します。
$ cd AMTK_AMSR2_1.11

3-3　コンパイル
以下のコマンドを順番に実行して、コンパイルを行います。
--with-hdf-include=には hdf5 のインクルードディレクトリ、
--with-hdf-lib=には hdf5 のライブラリディレクトリを指定します。
```
$ ./configure --with-hdf-include=/home/user1/utilhdf5_1.8.4-patch1/include
              --with-hdf-lib=/home/user1/utilhdf5_1.8.4-patch1/lib
$ make
```

3-4　環境設定
うるう秒ファイル leapsec.dat と物理量ファイル geophysical_file の場所を
環境変数に設定します。
この 2 つのファイルは AMTK ライブラリディレクトリ配下の share/data にあります。
以下のような設定をユーザーログインスクリプトなどに追加してください。

＜csh/tcsh 環境の場合 → .login ファイルまたは.cshrc ファイル＞
```
setenv AMSR2_LEAP_DATA /home/user1/util/AMTK_AMSR2_1.11/share/data/leapsec.dat
setenv GEOPHYSICALFILE /home/user1/util/AMTK_AMSR2_1.11/share/data/geophysical_file
```

＜sh/bash 環境の場合 → .profile ファイルまたは.bashrc ファイル＞
```
export AMSR2_LEAP_DATA=/home/user1/util/AMTK_AMSR2_1.11/share/data/leapsec.dat
export GEOPHYSICALFILE=/home/user1/util/AMTK_AMSR2_1.11/share/data/geophysical_file
```

(4)うるう秒ファイルの配置

GCOM-W1 データ提供サービスホームページ(http://gcom-w1.jaxa.jp/)から
うるう秒ファイル leapsec.dat をダウンロードします。
ダウンロードしたうるう秒ファイル leapsec.dat を適当なディレクトリに配置して、
ディレクトリを環境変数 AMSR2_LEAP_DATA に設定します。
以下のような設定をユーザーログインスクリプトなどに追加してください。

＜csh/tcsh 環境の場合 → .login ファイルまたは.cshrc ファイル＞
setenv AMSR2_LEAP_DATA /home/user1/util/leapsec.dat

＜sh/bash 環境の場合 → .profile ファイルまたは.bashrc ファイル＞
export AMSR2_LEAP_DATA=/home/user1/util/leapsec.dat

５．AMTK 編(C 言語)

赤文字の解説はサンプルプログラムについて説明しています。

青文字の解説は関数リファレンスまたは衛星基礎知識について説明しています。

５．１ L1B データ読込み

５．１．１ サンプルプログラム readL1B_amtk.c 解説

サンプルプログラム readL1B_amtk.c では、L1B データファイルから、以下のメタデータと格納データを読込んで、89G A ホーン緯度経度から低周波緯度経度を算出し、内容をテキスト表示します。

メタデータ	格納データ
* GeophysicalName - GranuleID - ObservationStartDateTime - EquatorCrossingDateTime - ObservationEndDateTime * NumberOfScans - OverlapScans - CoRegistrationParameterA1 - CoRegistrationParameterA2	* Scan Time * Latitude of Observation Point for 89A - Latitude of Observation Point for 89B - Longitude of Observation Point for 89A - Longitude of Observation Point for 89B * Brightness Temperature (6.9GHz,H) - Brightness Temperature (6.9GHz,V) - Brightness Temperature (7.3GHz,H) - Brightness Temperature (7.3GHz,V) - Brightness Temperature (10.7GHz,H) - Brightness Temperature (10.7GHz,V) - Brightness Temperature (18.7GHz,H) - Brightness Temperature (18.7GHz,V) - Brightness Temperature (23.8GHz,H) - Brightness Temperature (23.8GHz,V) - Brightness Temperature (36.5GHz,H) - Brightness Temperature (36.5GHz,V) - Brightness Temperature (89.0GHz-A,H) - Brightness Temperature (89.0GHz-A,V) - Brightness Temperature (89.0GHz-B,H) - Brightness Temperature (89.0GHz-B,V) * Pixel Data Quality 6 to 36 - Pixel Data Quality 89 * Land_Ocean Flag 6 to 36 - Land_Ocean Flag 89 * Earth Incidence - Earth Azimuth
89G A ホーン緯度経度から算出するデータ - 緯度経度(低周波平均) - 緯度経度(6G) - 緯度経度(7G) - 緯度経度(10G) - 緯度経度(18G) - 緯度経度(23G) - 緯度経度(36G) → P.15 基礎知識編３．１０参照	

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の AMTK 関数は、最初に使用する際に使用説明を記載します。

メタデータ読込みには、AMTK_getMetaDataName()関数を使用します。
配列データ読込みには、出力されるデータ型によって以下の 4 つの関数を使い分けます。

- 出力が時刻構造体の場合　→ AMTK_getScanTime()関数
- 出力が緯度経度構造体の場合 → AMTK_getLatLon()関数
- 出力が float 型の場合　→ AMTK_get_SwathFloat()関数
- 出力が int 型の場合　→ AMTK_get_SwathInt()関数

読込むデータ種類の指定は、アクセスラベルで行います。アクセスラベルとは、データ種類を指定するために AMTK で定義されている定数名です。

◇変数宣言　※左側の数字は行番号です

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include "AMTK.h"
 4
 5 // fixed value
 6 #define LMT 2200 // limit of NumberOfScans
 7
 8 int main(int argc, char *argv[]){
 9   // interface variable
10   int i,j;       // loop variable
11   int ret;       // return status
12   char buf[512]; // text buffer
13   void *vpnt;    // pointer to void
14   char *fn;      // filename
15   hid_t hnd;     // file handle
16
17   // meta data
18   char geo[512];  // GeophysicalName
19   char gid[512];  // GranuleID
20   char tm1[512];  // ObservationStartDateTime
21   char tm2[512];  // EquatorCrossingDateTime
22   char tm3[512];  // ObservationEndDateTime
23   int num;       // NumberOfScans
24   int ovr;       // OverlapScans
25   char prm1[512]; // CoRegistrationParameterA1
26   char prm2[512]; // CoRegistrationParameterA2
 :
28   // array data
29   AM2_COMMON_SCANTIME st[LMT]; // scantime
30   AM2_COMMON_LATLON ll89a[LMT][AM2_DEF_SNUM_HI]; // latlon for 89a
31   AM2_COMMON_LATLON ll89b[LMT][AM2_DEF_SNUM_HI]; // latlon for 89b
 :
39   float tb06h [LMT][AM2_DEF_SNUM_LO]; // tb for 06h
40   float tb06v [LMT][AM2_DEF_SNUM_LO]; // tb for 06v
 :
56   unsigned char pdq06h [LMT][AM2_DEF_SNUM_LO];    // pixel data quality for 06h
57   unsigned char pdq06v [LMT][AM2_DEF_SNUM_LO];    // pixel data quality for 06v
 :
66   unsigned char lof06 [LMT  ][AM2_DEF_SNUM_LO]; // land ocean flag for 06
67   unsigned char lof07 [LMT  ][AM2_DEF_SNUM_LO]; // land ocean flag for 07

75   float ear_in[AM2_DEF_SNUM_LO][LMT]; // earth incidence
76   float ear_az[AM2_DEF_SNUM_LO][LMT]; // earth azimuth
```

C 言語用 AMTK ヘッダーをインクルードします。

標準プロダクトのスキャン数上限は 2200 で充分ですが、
準リアルプロダクトを使用する場合は、
2 周回程度繋がる場合があるので、
LMT=9000 としてください。(標準の 4 パス分程度)

AMTK 用インターフェイス変数を宣言します。

メタデータ用変数を宣言します。

物理量名
グラニュール ID
観測開始時刻
赤道通過時刻
観測終了時刻
スキャン数
オーバーラップスキャン数
相対レジストレーション係数 A1
相対レジストレーション係数 A2

時刻データには AM2_COMMON_SCANTIME 構造体を使います。
緯度経度データには AM2_COMMON_LATLON 構造体を使います。
配列の次元数はデータによって異なります。
スキャン数はパスによって若干上下するので、上限を決めて宣言しておきます。
ここでは上限として LMT=2200 を設定しています。
AM2_DEF_SNUM_HI は AMTK に定義されている定数で、高解像度ピクセル数(486)です。
AM2_DEF_SNUM_LO は AMTK に定義されている定数で、低解像度ピクセル数(243)です。

時刻
緯度経度
輝度温度
品質フラグ
陸海フラグ
観測入射角
観測方位角

配列データ用変数を宣言します。

◇開始処理　※左側の数字は行番号です

ファイルをオープンします。

hnd=AMTK_openH5(fn)
fn: オープンするファイル名を指定します
hnd: [戻り値]成功の場合はファイルハンドル値、失敗の場合は負の値

```
 88   // open
 89   hnd=AMTK_openH5(fn);
 90   if(hnd<0){
 91     printf("AMTK_openH5 error: %s¥n", fn);
 92     printf("amtk status: %d¥n", hnd);
 93     exit(1);
 94   }
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込みます。

ret=AMTK_getMetaDataName(hnd,met,out)
hnd: ファイルハンドル値を指定します
met: メタデータ名称を指定します
out: メタデータ内容が返されます
ret: [戻り値]成功の場合は読込んだメタデータの文字数、失敗の場合は負の値

AMTK では出力変数はすべてポインタへのポインタとして渡します。
まずポインタを作ってから、そのポインタを渡します。

```
 96   // read meta: GeophysicalName
 97   vpnt=geo;
 98   ret=AMTK_getMetaDataName(hnd,"GeophysicalName",(char **)&vpnt);
 99   if(ret<0){
100     printf("AMTK_getMetaDataName error: GeophysicalName¥n");
101     printf("amtk status: %d¥n",ret);
102     exit(1);
103   }
104   printf("GeophysicalName: %s¥n",geo);
  :
```

コンパイラからの警告を避けるために、
void ポインタを適切な型にキャストします。

メタデータからスキャン数を読込みます。
配列データ読込みではスキャン数の指定が必要になるので、ここで読込んでおきます。

```
146   // read meta: NumberOfScans
147   vpnt=buf;
148   ret=AMTK_getMetaDataName(hnd,"NumberOfScans",(char **)&vpnt);
149   if(ret<0){
150     printf("AMTK_getMetaDataName error: NumberOfScans¥n");
151     printf("amtk status: %d¥n",ret);
152     exit(1);
153   }
154   num=atoi(buf);
155   printf("NumberOfScans: %d¥n",num);
```

メタデータは全て文字として取得されるので、
数値に変換しておきます。

◇時刻データ読込み　※左側の数字は行番号です

P.11 基礎知識編３．６参照

時刻データを読込みます。
時刻データは AM2_COMMON_SCANTIME 構造体の 1 次元配列で、サイズはスキャン数です。
時刻データの読込みには、AMTK_getScanTime()関数を使用します。

ret=AMTK_getScanTime(hnd,bgn,end,out)
hnd: ファイルハンドル値を指定します
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
out: 出力データが返されます
ret: [戻り値]失敗の場合は負の値

```
197   // read array: scantime
198   vpnt=st;
199   ret=AMTK_getScanTime(hnd,1,num,(AM2_COMMON_SCANTIME **)&vpnt);
200   if(ret<0){
201     printf("AMTK_getScanTime error.¥n");
202     printf("amtk status: %d¥n",ret);
203     exit(1);
204   }
205   printf("time[scan=0]: %04d/%02d/%02d %02d:%02d:%02d¥n"
206   , st[0].year
207   , st[0].month
208   , st[0].day
209   , st[0].hour
210   , st[0].minute
211   , st[0].second
212   );
```

◇緯度経度データ読込み　※左側の数字は行番号です

P.15 基礎知識編３．１０参照

緯度経度データを読込みます。
緯度経度データは AM2_COMMON_LATLON 構造体の 2 次元配列で、サイズはピクセル数×スキャン数です。
緯度経度データの読込みには、AMTK_getLatLon()関数を使用します。

ret=AMTK_getLatLon(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
214   // read array: latlon for 89a
215   vpnt=ll89a;
216   ret=AMTK_getLatLon(hnd,(AM2_COMMON_LATLON **)&vpnt,1,num,AM2_LATLON_89A);
217   if(ret<0){
218     printf("AMTK_getLatLon error: AM2_LATLON_89A¥n");
219     printf("amtk status: %d¥n",ret);
220     exit(1);
221   }
222   printf("latlon89a[scan=0][pixel=0]: (%9.4f,%9.4f)¥n", ll89a[0][0].lat,
ll89a[0][0].lon);
```

◇輝度温度データ読込み　※左側の数字は行番号です

輝度温度データを読込みます。
輝度温度データは float 型の 2 次元配列で、サイズはピクセル数×スキャン数です。
出力が float 型なので、AMTK_get_SwathFloat()関数を使用します。

ret=AMTK_get_SwathFloat(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
304   // read array: tb for 06h
305   vpnt=tb06h;
306   ret=AMTK_get_SwathFloat(hnd,(float **)&vpnt,1,num,AM2_TB06H);
307   if(ret<0){
308     printf("AMTK_get_SwathFloat error: AM2_TB06H¥n");
309     printf("amtk status: %d¥n",ret);
310     exit(1);
311   }
312   printf("tb06h[scan=0][pixel=0]: %9.2f¥n", tb06h[0][0]);
```

◇L1 品質フラグデータ読込み　※左側の数字は行番号です

L1 品質フラグデータを読込みます。
L1 品質フラグデータは int 型の 2 次元配列で、
L1 低解像度品質フラグデータのサイズは(ピクセル数*16)×スキャン数です。
L1 高解像度品質フラグデータのサイズは(ピクセル数* 8)×スキャン数です。
出力が int 型なので、AMTK_get_SwathInt()関数を使用します。

ret=AMTK_get_SwathInt(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
464   // read array: pixel data quality for low
465   vpnt=pdqlo;
466   ret=AMTK_get_SwathInt(hnd,(int **)&vpnt,1,num,AM2_PIX_QUAL_LO);
467   if(ret<0){
468     printf("AMTK_get_SwathInt error: AM2_PIX_QUAL_LO¥n");
469     printf("amtk status: %d¥n",ret);
470     exit(1);
471   }
```

L1 品質フラグデータは、各周波数（6GHz、7GHz)、偏波に対する RFI 情報がまとまって取得されます。
まとまったままでは扱い難いので、各周波数毎の配列に分割します。
格納値は 2 ビットのマスク値なので、各周波数毎に unsigned char 型の 2 次元配列を用意します。
L1 品質フラグデータには RFI(Radio Frequency Interference; 電波干渉)情報が格納されています。
電波干渉の発生がない場合は 00、発生の可能性がある場合は 10、発生がある場合は 11 となります。

```
472   for(j=0;j<num;++j){
473     for(i=0;i<AM2_DEF_SNUM_LO;++i){
474       pdq06v[j][i]=pdqlo[j][i*16+ 1]*10+pdqlo[j][i*16+ 0];
475       pdq06h[j][i]=pdqlo[j][i*16+ 3]*10+pdqlo[j][i*16+ 2];
476       pdq07v[j][i]=pdqlo[j][i*16+ 5]*10+pdqlo[j][i*16+ 4];
477       pdq07h[j][i]=pdqlo[j][i*16+ 7]*10+pdqlo[j][i*16+ 6];
478     }
479   }
```

◇L1B 陸海フラグデータ読込み　※左側の数字は行番号です

陸海フラグデータを読込みます。
陸海フラグデータは int 型の 2 次元配列で、
L1B 低解像度陸海データのサイズはピクセル数×(スキャン数*6)です。
L1B 高解像度陸海データのサイズはピクセル数×(スキャン数*2)です。
出力が int 型なので、AMTK_get_SwathInt()関数を使用します。

P. 14 基礎知識編３． ９参照

```
506   // read array: land ocean flag for low
507   vpnt=loflo;
508   ret=AMTK_get_SwathInt(hnd,(int **)&vpnt,1,num,AM2_LOF_LO);
509   if(ret<0){
510     printf("AMTK_get_SwathInt error: AM2_LOF_LO¥n");
511     printf("amtk status: %d¥n",ret);
512     exit(1);
513   }
```

陸海フラグデータは、解像度毎に全周波数データがまとまって取得されます。
まとまったままでは扱い難いので、各周波数毎の配列に分割します。
格納値は 0～100 のフラグ値なので、各周波数毎に unsigned char 型の 2 次元配列を用意します。

```
514   for(j=0;j<num;++j){
515     for(i=0;i<AM2_DEF_SNUM_LO;++i){
516       lof06[j][i]=loflo[num*0+j][i];
517       lof07[j][i]=loflo[num*1+j][i];
518       lof10[j][i]=loflo[num*2+j][i];
519       lof18[j][i]=loflo[num*3+j][i];
520       lof23[j][i]=loflo[num*4+j][i];
521       lof36[j][i]=loflo[num*5+j][i];
522     }
523   }
```

◇観測入射角データ読込み　※左側の数字は行番号です

観測入射角データを読込みます。
観測入射角データは float 型の 2 次元配列で、サイズはピクセル数×スキャン数です。
出力が float 型なので、AMTK_get_SwathFloat()関数を使用します。

```
548   // read array: earth incidence
549   vpnt=ear_in;
550   ret=AMTK_get_SwathFloat(hnd,(float **)&vpnt,1,num,AM2_EARTH_INC);
551   if(ret<0){
552     printf("AMTK_get_SwathFloat error: AM2_EARTH_INC¥n");
553     printf("amtk status: %d¥n",ret);
554     exit(1);
555   }
556   printf("ear_in[scan=0][pixel=0]: %9.2f¥n",ear_in[0][0]);
```

◇終了処理　※左側の数字は行番号です

ファイルをクローズします。

ret=AMTK_closeH5(hnd)
hnd: クローズするファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

```
568   // close
569   ret=AMTK_closeH5(hnd);
```

５．１．２ コンパイル方法(build_readL1B_amtk_c.sh 解説)

コンパイルに使用するスクリプト build_readL1B_amtk_c.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～9 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 AMTK=/home/user1/util/AMTK_AMSR2_1.11
 8 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 9 SZIP=/home/user1/util/szip_2.1
10
```

12 行目には使用するコンパイラを指定します。
インテルコンパイラ(icc)または PG コンパイラ(pgcc)または GNU コンパイラ(gcc)を指定します。

```
11 # compiler
12 cc=icc
13
14 # source filename
15 csrc=readL1B_amtk.c
16
17 # output filename
18 out=readL1B_amtk_c
19
20 # library order
21 lib="-lAMSR2 -lhdf5 -lsz -lz -lm"
22
23 # compile
24 cmd="$cc -g $csrc -o $out -I$AMTK/include -I$HDF5/include -I$SZIP/include
-L$AMTK/lib -L$HDF5/lib -L$SZIP/lib $lib"
25 echo $cmd
26 $cmd
27
28 # garbage
29 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL1B_amtk_c.sh
icc -g readL1B_amtk.c -o readL1B_amtk_c
-I/home/user1/util/AMTK_AMSR2_1.11/include
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/AMTK_AMSR2_1.11/lib
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lAMSR2 -lhdf5 -lsz -lz -lm
```

５．１．３ プログラム実行結果のサンプル

サンプルプログラムでは固定配列を多数使用しているため、環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞
$ unlimit

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、4 つとも実行してください。
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited

```
$ ./readL1B_amtk_c GW1AM2_201207261145_055A_L1SGBTBR_0000000.h5
input file: GW1AM2_201207261145_055A_L1SGBTBR_0000000.h5
GeophysicalName: Brightness Temperature
GranuleID: GW1AM2_201207261145_055A_L1SGBTBR_0000000
ObservationStartDateTime: 2012-07-26T11:45:43.018Z
EquatorCrossingDateTime: 2012-07-26T12:12:37.848Z
ObservationEndDateTime: 2012-07-26T12:35:09.735Z
NumberOfScans: 1979
limit of NumberOfScans = 2200
OverlapScans: 20
CoRegistrationParameterA1: 6G-1.25000,7G-1.00000,10G-1.25000,18G-1.25000,23G-1.25000,36G-1.00000
CoRegistrationParameterA2: 6G-0.00000,7G--0.10000,10G--0.25000,18G-0.00000,23G--0.25000,36G-0.00000
time[scan=0]: 2012/07/26 11:45:43
latlon89a[scan=0][pixel=0]: ( -73.3289, 136.7714)
latlon89b[scan=0][pixel=0]: ( -73.4038, 137.1498)
latlonlm[scan=0][pixel=0]: ( -73.3538, 136.6228)
latlon06[scan=0][pixel=0]: ( -73.3592, 136.6213)
latlon07[scan=0][pixel=0]: ( -73.3497, 136.6429)
latlon10[scan=0][pixel=0]: ( -73.3506, 136.6001)
latlon18[scan=0][pixel=0]: ( -73.3592, 136.6213)
latlon23[scan=0][pixel=0]: ( -73.3506, 136.6001)
latlon36[scan=0][pixel=0]: ( -73.3532, 136.6514)
tb06h[scan=0][pixel=0]:    173.28
tb06v[scan=0][pixel=0]:    208.22
tb07h[scan=0][pixel=0]:    173.07
tb07v[scan=0][pixel=0]:    207.54
tb10h[scan=0][pixel=0]:    170.94
tb10v[scan=0][pixel=0]:    204.95
tb18h[scan=0][pixel=0]:    164.85
tb18v[scan=0][pixel=0]:    199.84
tb23h[scan=0][pixel=0]:    163.22
tb23v[scan=0][pixel=0]:    196.53
tb36h[scan=0][pixel=0]:    156.56
tb36v[scan=0][pixel=0]:    186.39
tb89ah[scan=0][pixel=0]:    163.76
tb89av[scan=0][pixel=0]:    179.27
tb89bh[scan=0][pixel=0]:    170.60
tb89bv[scan=0][pixel=0]:    188.16
```

```
pdq06h[scan=0][pixel=0]):    0
pdq06v[scan=0][pixel=0]):    0
pdq07h[scan=0][pixel=0]):    0
pdq07v[scan=0][pixel=0]):    0
pdq10h[scan=0][pixel=0]):    0
pdq10v[scan=0][pixel=0]):    0
pdq18h[scan=0][pixel=0]):    0
pdq18v[scan=0][pixel=0]):    0
pdq23h[scan=0][pixel=0]):    0
pdq23v[scan=0][pixel=0]):    0
pdq36h[scan=0][pixel=0]):    0
pdq36v[scan=0][pixel=0]):    0
pdq89ah[scan=0][pixel=0]):    0
pdq89av[scan=0][pixel=0]):    0
pdq89ah[scan=0][pixel=0]):    0
pdq89av[scan=0][pixel=0]):    0
lof06[scan=0][pixel=0]: 100
lof07[scan=0][pixel=0]: 100
lof10[scan=0][pixel=0]: 100
lof18[scan=0][pixel=0]: 100
lof23[scan=0][pixel=0]: 100
lof36[scan=0][pixel=0]: 100
lof89a[scan=0][pixel=0]: 100
lof89b[scan=0][pixel=0]: 100
ear_in[scan=0][pixel=0]:     55.20
ear_az[scan=0][pixel=0]:    144.76
```

５．２ L1R データ読込み

５．２．１ サンプルプログラム readL1R_amtk.c 解説

サンプルプログラム readL1R_amtk.c では、L1R データファイルから、以下のメタデータと格納データを読込んで、89G A ホーン緯度経度から低周波用緯度経度を抽出し、内容をテキスト表示します。
このサンプルプログラムでは、格納されている L1R 輝度温度の一部のみ取扱います。L1R 輝度温度データ一覧については、P.14「３．８　L1 リサンプリングデータ」を参照ください。

メタデータ	格納データ
* GeophysicalName - GranuleID - ObservationStartDateTime - EquatorCrossingDateTime - ObservationEndDateTime * NumberOfScans - OverlapScans - CoRegistrationParameterA1 - CoRegistrationParameterA2	* Scan Time * Latitude of Observation Point for 89A - Latitude of Observation Point for 89B - Longitude of Observation Point for 89A - Longitude of Observation Point for 89B * Brightness Temperature (res06,6.9GHz,H) - Brightness Temperature (res06,6.9GHz,V) - Brightness Temperature (res06,7.3GHz,H) - Brightness Temperature (res06,7.3GHz,V) - Brightness Temperature (res10,10.7GHz,H) - Brightness Temperature (res10,10.7GHz,V) - Brightness Temperature (res23,18.7GHz,H) - Brightness Temperature (res23,18.7GHz,V) - Brightness Temperature (res23,23.8GHz,H) - Brightness Temperature (res23,23.8GHz,V) - Brightness Temperature (res36,36.5GHz,H) - Brightness Temperature (res36,36.5GHz,V) - Brightness Temperature (res36,89.0GHz,H) - Brightness Temperature (res36,89.0GHz,V) - Brightness Temperature (original,89GHz-A,H) - Brightness Temperature (original,89GHz-A,V) - Brightness Temperature (original,89GHz-B,H) - Brightness Temperature (original,89GHz-B,V) * Pixel Data Quality 6 to 36 - Pixel Data Quality 89 * Land_Ocean Flag 6 to 36 - Land_Ocean Flag 89 * Earth Incidence - Earth Azimuth
89G A ホーン緯度経度から抽出するデータ	
- 低周波用緯度経度 → P.15 基礎知識編３．１０参照	

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の AMTK 関数は、最初に使用する際に使用説明を記載します。

メタデータ読込みには、AMTK_getMetaDataName()関数を使用します。
配列データ読込みには、出力されるデータ型によって以下の 4 つの関数を使い分けます。

- 出力が時刻構造体の場合　→ AMTK_getScanTime()関数
- 出力が緯度経度構造体の場合 → AMTK_getLatLon()関数
- 出力が float 型の場合　→ AMTK_get_SwathFloat()関数
- 出力が int 型の場合　→ AMTK_get_SwathInt()関数

読込むデータ種類の指定は、アクセスラベルで行います。アクセスラベルとは、データ種類を指定するために AMTK で定義されている定数名です。

◇変数宣言　※左側の数字は行番号です

```
  1 #include <stdio.h>
  2 #include <stdlib.h>
  3 #include "AMTK.h"
  4
  5 // fixed value
  6 #define LMT 2200 // limit of NumberOfScans
  7
  8 int main(int argc, char *argv[]){
  9   // interface variable
 10   int i,j;       // loop variable
 11   int ret;       // return status
 12   char buf[512]; // text buffer
 13   void *vpnt;    // pointer to void
 14   char *fn;      // filename
 15   hid_t hnd;     // file handle
 16
 17   // meta data
 18   char geo[512];  // GeophysicalName
 19   char gid[512];  // GranuleID
 20   char tm1[512];  // ObservationStartDateTime
 21   char tm2[512];  // EquatorCrossingDateTime
 22   char tm3[512];  // ObservationEndDateTime
 23   int num;       // NumberOfScans
 24   int ovr;       // OverlapScans
 25   char prm1[512]; // CoRegistrationParameterA1
 26   char prm2[512]; // CoRegistrationParameterA2

 28   // array data
 29   AM2_COMMON_SCANTIME st[LMT]; // scantime
 30   AM2_COMMON_LATLON ll89ar[LMT][AM2_DEF_SNUM_HI]; // latlon for 89a altitude
revised
 31   AM2_COMMON_LATLON ll89br[LMT][AM2_DEF_SNUM_HI]; // latlon for 89b altitude
revised
  :
 33   float tb06h06 [LMT][AM2_DEF_SNUM_LO]; // tb for 06h, resolution 06G
 34   float tb06v06 [LMT][AM2_DEF_SNUM_LO]; // tb for 06v, resolution
  :
 52   unsigned char pdq06h [LMT][AM2_DEF_SNUM_LO];    // pixel data qual
 53   unsigned char pdq06v [LMT][AM2_DEF_SNUM_LO];    // pixel data quality for 06v
  :
 62   unsigned char lof06 [LMT  ][AM2_DEF_SNUM_LO]; // land ocean flag for 06
 63   unsigned char lof10 [LMT  ][AM2_DEF_SNUM_LO]; // land ocean flag for 10
  :
 69   float ear_in[AM2_DEF_SNUM_LO][LMT]; // earth incidence
 70   float ear_az[AM2_DEF_SNUM_LO][LMT]; // earth azimuth
```

C 言語用 AMTK ヘッダーをインクルードします。

標準プロダクトのスキャン数上限は 2200 で充分ですが、
準リアルプロダクトを使用する場合は、
2 周回程度繋がる場合があるので、
LMT=9000 としてください。(標準の 4 パス分程度)

AMTK 用インターフェイス変数を宣言します。

メタデータ用変数を宣言します。

物理量名
グラニュール ID
観測開始時刻
赤道通過時刻
観測終了時刻
スキャン数
オーバーラップスキャン数
相対レジストレーション係数 A1
相対レジストレーション係数 A2

時刻データには AM2_COMMON_SCANTIME 構造体を使います。
緯度経度データには AM2_COMMON_LATLON 構造体を使います。
配列の次元数はデータによって異なります。
スキャン数はパスによって若干上下するので、上限を決めて宣言しておきます。
ここでは上限として LMT=2200 を設定しています。
AM2_DEF_SNUM_HI は AMTK に定義されている定数で、高解像度ピクセル数(486)です。
AM2_DEF_SNUM_LO は AMTK に定義されている定数で、低解像度ピクセル数(243)です。

時刻
緯度経度
輝度温度
品質フラグ
陸海フラグ
観測入射角
観測方位角

配列データ用変数を宣言します。

◇開始処理　※左側の数字は行番号です

ファイルをオープンします。

hnd=AMTK_openH5(fn)
fn: オープンするファイル名を指定します
hnd: [戻り値]成功の場合はファイルハンドル値、失敗の場合は負の値

```
 82   // open
 83   hnd=AMTK_openH5(fn);
 84   if(hnd<0){
 85     printf("AMTK_openH5 error: %s¥n", fn);
 86     printf("amtk status: %d¥n", hnd);
 87     exit(1);
 88   }
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込みます。

ret=AMTK_getMetaDataName(hnd,met,out)
hnd: ファイルハンドル値を指定します
met: メタデータ名称を指定します
out: メタデータ内容が返されます
ret: [戻り値]成功の場合は読込んだメタデータの文字数、失敗の場合は負の値

AMTK では出力変数はすべてポインタへのポインタとして渡します。
まずポインタを作ってから、そのポインタを渡します。

```
 90   // read meta: GeophysicalName
 91   vpnt=geo;
 92   ret=AMTK_getMetaDataName(hnd,"GeophysicalName",(char **)&vpnt);
 93   if(ret<0){
 94     printf("AMTK_getMetaDataName error: GeophysicalName¥n");
 95     printf("amtk status: %d¥n",ret);
 96     exit(1);
 97   }
 98   printf("GeophysicalName: %s¥n",geo);
```

コンパイラからの警告を避けるために、
void ポインタを適切な型にキャストします。

メタデータからスキャン数を読込みます。
配列データ読込みではスキャン数の指定が必要になるので、ここで読込んでおきます。

```
140   // read meta: NumberOfScans
141   vpnt=buf;
142   ret=AMTK_getMetaDataName(hnd,"NumberOfScans",(char **)&vpnt);
143   if(ret<0){
144     printf("AMTK_getMetaDataName error: NumberOfScans¥n");
145     printf("amtk status: %d¥n",ret);
146     exit(1);
147   }
148   num=atoi(buf);
149   printf("NumberOfScans: %d¥n",num);
```

メタデータは全て文字として取得されるので、
数値に変換しておきます。

◇時刻データ読込み　※左側の数字は行番号です

P.11 基礎知識編３． ６参照

時刻データを読込みます。
時刻データは AM2_COMMON_SCANTIME 構造体の 1 次元配列で、サイズはスキャン数です。
時刻データの読込みには、AMTK_getScanTime()関数を使用します。

ret=AMTK_getScanTime(hnd,bgn,end,out)
hnd: ファイルハンドル値を指定します
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
out: 出力データが返されます
ret: [戻り値]失敗の場合は負の値

```
191   // read array: scantime
192   vpnt=st;
193   ret=AMTK_getScanTime(hnd,1,num,(AM2_COMMON_SCANTIME **)&vpnt);
194   if(ret<0){
195     printf("AMTK_getScanTime error.¥n");
196     printf("amtk status: %d¥n",ret);
197     exit(1);
198   }
199   printf("time[scan=0]: %04d/%02d/%02d %02d:%02d:%02d¥n"
200   , st[0].year
201   , st[0].month
202   , st[0].day
203   , st[0].hour
204   , st[0].minute
205   , st[0].second
206   );
```

◇緯度経度データ読込み　※左側の数字は行番号です

P.15 基礎知識編３． １０参照

緯度経度データを読込みます。
緯度経度データは AM2_COMMON_LATLON 構造体の 2 次元配列で、サイズはピクセル数×スキャン数です。
緯度経度データの読込みには、AMTK_getLatLon()関数を使用します。

ret=AMTK_getLatLon(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
208   // read array: latlon for 89a altitude revised
209   vpnt=ll89ar;
210   ret=AMTK_getLatLon(hnd,(AM2_COMMON_LATLON **)&vpnt,1,num
,AM2_LATLON_RS_89A);
211   if(ret<0){
212     printf("AMTK_getLatLon error: AM2_LATLON_RS_89A¥n");
213     printf("amtk status: %d¥n",ret);
214     exit(1);
215   }
216   printf("latlon89ar[scan=0][pixel=0]: (%9.4f,%9.4f)¥n", ll89ar[0][0].lat,
ll89ar[0][0].lon);
```

◇輝度温度データ読込み　※左側の数字は行番号です

輝度温度データを読込みます。
輝度温度データは float 型の 2 次元配列で、サイズはピクセル数×スキャン数です。
出力が float 型なので、AMTK_get_SwathFloat()関数を使用します。

ret=AMTK_get_SwathFloat(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
238   // read array: tb for 06h, resolution 06G
239   vpnt=tb06h06;
240   ret=AMTK_get_SwathFloat(hnd,(float **)&vpnt,1,num,AM2_RES06_TB06H);
241   if(ret<0){
242     printf("AMTK_get_SwathFloat error: AM2_RES06_TB06H¥n");
243     printf("amtk status: %d¥n",ret);
244     exit(1);
245   }
246   printf("tb06h06[scan=0][pixel=0]: %9.2f¥n", tb06h06[0][0]);
```

◇L1 品質フラグデータ読込み　※左側の数字は行番号です

L1 品質フラグデータを読込みます。
L1 品質フラグデータは int 型の 2 次元配列で、
L1 低解像度品質フラグデータのサイズは(ピクセル数*16)×スキャン数です。
L1 高解像度品質フラグデータのサイズは(ピクセル数* 8)×スキャン数です。
出力が int 型なので、AMTK_get_SwathInt()関数を使用します。

ret=AMTK_get_SwathInt(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
418   // read array: pixel data quality for low
419   vpnt=pdqlo;
420   ret=AMTK_get_SwathInt(hnd,(int **)&vpnt,1,num,AM2_PIX_QUAL_LO);
421   if(ret<0){
422     printf("AMTK_get_SwathInt error: AM2_PIX_QUAL_LO¥n");
423     printf("amtk status: %d¥n",ret);
424     exit(1);
425   }
```

L1 品質フラグデータは、各周波数（6GHz、7GHz)、偏波に対する RFI 情報がまとまって取得されます。
まとまったままでは扱い難いので、各周波数毎の配列に分割します。
格納値は 2 ビットのマスク値なので、各周波数毎に unsigned char 型の 2 次元配列を用意します。
L1 品質フラグデータには RFI(Radio Frequency Interference; 電波干渉)情報が格納されています。
電波干渉の発生がない場合は 00、発生の可能性がある場合は 10、発生がある場合は 11 となります。

```
426   for(j=0;j<num;++j){
427     for(i=0;i<AM2_DEF_SNUM_LO;++i){
428       pdq06v[j][i]=pdqlo[j][i*16+ 1]*10+pdqlo[j][i*16+ 0];
429       pdq06h[j][i]=pdqlo[j][i*16+ 3]*10+pdqlo[j][i*16+ 2];
430       pdq07v[j][i]=pdqlo[j][i*16+ 5]*10+pdqlo[j][i*16+ 4];
431       pdq07h[j][i]=pdqlo[j][i*16+ 7]*10+pdqlo[j][i*16+ 6];
432     }
433   }
```

◇L1R 陸海フラグデータ読込み　※左側の数字は行番号です

陸海フラグデータを読込みます。
陸海フラグデータは int 型の 2 次元配列で、
L1R 低解像度陸海データのサイズはピクセル数×(スキャン数*4)です。
L1R 高解像度陸海データのサイズはピクセル数×(スキャン数*2)です。
出力が int 型なので、AMTK_get_SwathInt()関数を使用します。

→ P. 14 基礎知識編３． ９参照

```
460   // read array: land ocean flag for low
461   vpnt=loflo;
462   ret=AMTK_get_SwathInt(hnd,(int **)&vpnt,1,num,AM2_LOF_RES_LO);
463   if(ret<0){
464     printf("AMTK_get_SwathInt error: AM2_LOF_RES_LO¥n");
465     printf("amtk status: %d¥n",ret);
466     exit(1);
467   }
```

陸海フラグデータは、解像度毎に全周波数データがまとまって取得されます。
まとまったままでは扱い難いので、各周波数毎の配列に分割します。
格納値は 0～100 のフラグ値なので、各周波数毎に unsigned char 型の 2 次元配列を用意します。

```
468   for(j=0;j<num;++j){
469     for(i=0;i<AM2_DEF_SNUM_LO;++i){
470       lof06[j][i]=loflo[num*0+j][i];
471       lof10[j][i]=loflo[num*1+j][i];
472       lof23[j][i]=loflo[num*2+j][i];
473       lof36[j][i]=loflo[num*3+j][i];
474     }
475   }
```

◇観測入射角データ読込み　※左側の数字は行番号です

観測入射角データを読込みます。
観測入射角データは float 型の 2 次元配列で、サイズはピクセル数×スキャン数です。
出力が float 型なので、AMTK_get_SwathFloat()関数を使用します。

```
498   // read array: earth incidence
499   vpnt=ear_in;
500   ret=AMTK_get_SwathFloat(hnd,(float **)&vpnt,1,num,AM2_EARTH_INC);
501   if(ret<0){
502     printf("AMTK_get_SwathFloat error: AM2_EARTH_INC¥n");
503     printf("amtk status: %d¥n",ret);
504     exit(1);
505   }
506   printf("ear_in[scan=0][pixel=0]: %9.2f¥n",ear_in[0][0]);
```

◇終了処理　※左側の数字は行番号です

ファイルをクローズします。

ret=AMTK_closeH5(hnd)
hnd: クローズするファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

```
518   // close
519   ret=AMTK_closeH5(hnd);
```

５．２．２ コンパイル方法(build_readL1R_amtk_c.sh 解説)

コンパイルに使用するスクリプト build_readL1R_amtk_c.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～9 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 AMTK=/home/user1/util/AMTK_AMSR2_1.11
 8 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 9 SZIP=/home/user1/util/szip_2.1
10
```

12 行目には使用するコンパイラを指定します。
インテルコンパイラ(icc)または PG コンパイラ(pgcc)または GNU コンパイラ(gcc)を指定します。

```
11 # compiler
12 cc=icc
13
14 # source filename
15 csrc=readL1R_amtk.c
16
17 # output filename
18 out=readL1R_amtk_c
19
20 # library order
21 lib="-lAMSR2 -lhdf5 -lsz -lz -lm"
22
23 # compile
24 cmd="$cc -g $csrc -o $out -I$AMTK/include -I$HDF5/include -I$SZIP/include
-L$AMTK/lib -L$HDF5/lib -L$SZIP/lib $lib"
25 echo $cmd
26 $cmd
27
28 # garbage
29 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL1R_amtk_c.sh
icc -g readL1R_amtk.c -o readL1R_amtk_c
-I/home/user1/util/AMTK_AMSR2_1.11/include
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/AMTK_AMSR2_1.11/lib
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lAMSR2 -lhdf5 -lsz -lz -lm
```

５．２．３ プログラム実行結果のサンプル

サンプルプログラムでは固定配列を多数使用しているため、環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞
$ unlimit

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、4 つとも実行してください。
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited

```
$ ./readL1R_amtk_c GW1AM2_201207261145_055A_L1SGRTBR_
0000000.h5
input file: GW1AM2_201207261145_055A_L1SGRTBR_000000
0.h5
GeophysicalName: Brightness Temperature
GranuleID: GW1AM2_201207261145_055A_L1SGRTBR_0000000
ObservationStartDateTime: 2012-07-26T11:45:43.018Z
EquatorCrossingDateTime: 2012-07-26T12:12:37.848Z
ObservationEndDateTime: 2012-07-26T12:35:09.735Z
NumberOfScans: 1979
limit of NumberOfScans = 2200
OverlapScans: 20
CoRegistrationParameterA1: 6G-0.00000,7G-0.00000,10G-
0.00000,18G-0.00000,23G-0.00000,36G-0.00000
CoRegistrationParameterA2: 6G-0.00000,7G-0.00000,10G-
0.00000,18G-0.00000,23G-0.00000,36G-0.00000
time[scan=0]: 2012/07/26 11:45:43
latlon89ar[scan=0][pixel=0]: ( -73.3581, 136.8432)
latlon89br[scan=0][pixel=0]: ( -73.4328, 137.2216)
latlonlr[scan=0][pixel=0]: ( -73.3581, 136.8432)
tb06h06[scan=0][pixel=0]:    173.41
tb06v06[scan=0][pixel=0]:    208.09
tb07h06[scan=0][pixel=0]:    173.07
tb07v06[scan=0][pixel=0]:    207.11
tb10h10[scan=0][pixel=0]:    170.40
tb10v10[scan=0][pixel=0]:    204.58
tb18h23[scan=0][pixel=0]:    165.83
tb18v23[scan=0][pixel=0]:    199.41
tb23h23[scan=0][pixel=0]:    163.55
tb23v23[scan=0][pixel=0]:    195.90
tb36h36[scan=0][pixel=0]:    153.55
tb36v36[scan=0][pixel=0]:    183.95
tb89h36[scan=0][pixel=0]:    163.86
tb89v36[scan=0][pixel=0]:    181.05
tb89ah[scan=0][pixel=0]:    163.76
tb89av[scan=0][pixel=0]:    179.27
tb89bh[scan=0][pixel=0]:    170.60
tb89bv[scan=0][pixel=0]:    188.16
```

```
pdq06h[scan=0][pixel=0]:   0
pdq06v[scan=0][pixel=0]:   0
pdq07h[scan=0][pixel=0]:   0
pdq07v[scan=0][pixel=0]:   0
pdq10h[scan=0][pixel=0]:   0
pdq10v[scan=0][pixel=0]:   0
pdq18h[scan=0][pixel=0]:   0
pdq18v[scan=0][pixel=0]:   0
pdq23h[scan=0][pixel=0]:   0
pdq23v[scan=0][pixel=0]:   0
pdq36h[scan=0][pixel=0]:   0
pdq36v[scan=0][pixel=0]:   0
pdq89ah[scan=0][pixel=0]:   0
pdq89av[scan=0][pixel=0]:   0
pdq89ah[scan=0][pixel=0]:   0
pdq89av[scan=0][pixel=0]:   0
lof06[scan=0][pixel=0]: 100
lof10[scan=0][pixel=0]: 100
lof23[scan=0][pixel=0]: 100
lof36[scan=0][pixel=0]: 100
lof89a[scan=0][pixel=0]: 100
lof89b[scan=0][pixel=0]: 100
ear_in[scan=0][pixel=0]:     55.20
ear_az[scan=0][pixel=0]:    144.76
```

５．３ L2 低解像度データ読込み

積算雲水量(CLW)・海氷密接度(SIC)・土壌水分量(SMC)・積雪深(SND)・海面水温(SST)・海上風速(SSW)・可降水量(TPW)が L2 低解像度です。降水量(PRC)は L2 高解像度を参照ください。

５．３．１ サンプルプログラム readL2L_amtk.c 解説

サンプルプログラム readL2L_amtk.c では、L2 低解像度データファイルから、以下のメタデータと格納データを読込んで、内容をテキスト表示します。

メタデータ	格納データ
* GeophysicalName - GranuleID - ObservationStartDateTime - EquatorCrossingDateTime - ObservationEndDateTime * NumberOfScans - OverlapScans	* Scan Time * Latitude of Observation Point * Longitude of Observation Point * Geophysical Data * Pixel Data Quality P.15 基礎知識編３．１０参照

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の AMTK 関数は、最初に使用する際に使用説明を記載します。

メタデータ読込みには、AMTK_getMetaDataName()関数を使用します。
配列データ読込みには、出力されるデータ型によって以下の 4 つの関数を使い分けます。

- 出力が時刻構造体の場合 → AMTK_getScanTime()関数
- 出力が緯度経度構造体の場合 → AMTK_getLatLon()関数
- 出力が float 型の場合 → AMTK_get_SwathFloat()関数
- 出力が unsigned char 型の場合 → AMTK_get_SwathUChar()関数

読込むデータ種類の指定は、アクセスラベルで行います。アクセスラベルとは、データ種類を指定するために AMTK で定義されている定数名です。

◇変数宣言　※左側の数字は行番号です

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <string.h>
 4 #include "AMTK.h"
 5
 6 // fixed value
 7 #define LMT 2200 // limit of NumberOfScans
 :
10 int main(int argc, char *argv[]){
11   // interface variable
12   int i,j;       // loop variable
13   int ret;       // return status
14   char buf[512]; // text buffer
15   void *vpnt;    // pointer to void
16   char *fn;      // filename
17   hid_t hnd;     // file handle
18
19   // meta data
20   char geo[512];  // GeophysicalName
21   char gid[512];  // GranuleID
22   char tm1[512];  // ObservationStartDateTime
23   char tm2[512];  // EquatorCrossingDateTime
24   char tm3[512];  // ObservationEndDateTime
25   int num;       // NumberOfScans
26   int ovr;       // OverlapScans
28   // array data
29   AM2_COMMON_SCANTIME st[LMT]; // scantime
30   AM2_COMMON_LATLON ll[LMT][AM2_DEF_SNUM_LO]; // latlon
31   float geo1[LMT][AM2_DEF_SNUM_LO]; // geophysical data layer 1
32   float geo2[LMT][AM2_DEF_SNUM_LO]; // geophysical data layer 2
33   unsigned char pdq1[LMT][AM2_DEF_SNUM_LO]; // pixel data quality layer 1
34   unsigned char pdq2[LMT][AM2_DEF_SNUM_LO]; // pixel data quality layer 2
35   unsigned char pdqtmp[LMT*2][AM2_DEF_SNUM_LO]; // pixel data quality temporary
```

C 言語用 AMTK ヘッダーをインクルードします。

標準プロダクトのスキャン数上限は 2200 で充分ですが、
準リアルプロダクトを使用する場合は、
2 周回程度繋がる場合があるので、
LMT=9000 としてください。(標準の 4 パス分程度)

AMTK 用インターフェイス変数を宣言します。

メタデータ用変数を宣言します。

物理量名
グラニュール ID
観測開始時刻
赤道通過時刻
観測終了時刻
スキャン数
オーバーラップスキャン数

時刻データには AM2_COMMON_SCANTIME 構造体を使います。
緯度経度データには AM2_COMMON_LATLON 構造体を使います。
配列の次元数はデータによって異なります。
スキャン数はパスによって若干上下するので、上限を決めて宣言しておきます。
ここでは上限として LMT=2200 を設定しています。
AM2_DEF_SNUM_HI は AMTK に定義されている定数で、高解像度ピクセル数(486)です。
AM2_DEF_SNUM_LO は AMTK に定義されている定数で、低解像度ピクセル数(243)です。

時刻
緯度経度
物理量
品質フラグ

配列データ用変数を宣言します。

◇開始処理　※左側の数字は行番号です

ファイルをオープンします。

hnd=AMTK_openH5(fn)
fn: オープンするファイル名を指定します
hnd: [戻り値]成功の場合はファイルハンドル値、失敗の場合は負の値

```
47   // open
48   hnd=AMTK_openH5(fn);
49   if(hnd<0){
50     printf("AMTK_openH5 error: %s¥n", fn);
51     printf("amtk status: %d¥n", hnd);
52     exit(1);
53   }
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込みます。

ret=AMTK_getMetaDataName(hnd,met,out)
hnd: ファイルハンドル値を指定します
met: メタデータ名称を指定します
out: メタデータ内容が返されます
ret: [戻り値]成功の場合は読込んだメタデータの文字数、失敗の場合は負の値

AMTK では出力変数はすべてポインタへのポインタとして渡します。
まずポインタを作ってから、そのポインタを渡します。

```
55   // read meta: GeophysicalName
56   vpnt=geo;
57   ret=AMTK_getMetaDataName(hnd,"GeophysicalName",(char **)&vpnt);
58   if(ret<0){
59     printf("AMTK_getMetaDataName error: GeophysicalName¥n");
60     printf("amtk status: %d¥n",ret);
61     exit(1);
62   }
63   printf("GeophysicalName: %s¥n",geo);
```

コンパイラからの警告を避けるために、
void ポインタを適切な型にキャストします。

メタデータからスキャン数を読込みます。
配列データ読込みではスキャン数の指定が必要になるので、ここで読込んでおきます。

```
118  // read meta: NumberOfScans
119  vpnt=buf;
120  ret=AMTK_getMetaDataName(hnd,"NumberOfScans",(char **)&vpnt);
121  if(ret<0){
122    printf("AMTK_getMetaDataName error: NumberOfScans¥n");
123    printf("amtk status: %d¥n",ret);
124    exit(1);
125  }
126  num=atoi(buf);
127  printf("NumberOfScans: %d¥n",num);
```

メタデータは全て文字として取得されるので、
数値に変換しておきます。

◇時刻データ読込み　※左側の数字は行番号です

P.11 基礎知識編３． ６参照

時刻データを読込みます。
時刻データは AM2_COMMON_SCANTIME 構造体の 1 次元配列で、サイズはスキャン数です。
時刻データの読込みには、AMTK_getScanTime()関数を使用します。

ret=AMTK_getScanTime(hnd,bgn,end,out)
hnd: ファイルハンドル値を指定します
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
out: 出力データが返されます
ret: [戻り値]失敗の場合は負の値

```
149   // read array: scantime
150   vpnt=st;
151   ret=AMTK_getScanTime(hnd,1,num,(AM2_COMMON_SCANTIME **)&vpnt);
152   if(ret<0){
153     printf("AMTK_getScanTime error.¥n");
154     printf("amtk status: %d¥n",ret);
155     exit(1);
156   }
157   printf("time[scan=0]: %04d/%02d/%02d %02d:%02d:%02d¥n"
158   , st[0].year
159   , st[0].month
160   , st[0].day
161   , st[0].hour
162   , st[0].minute
163   , st[0].second
164   );
```

◇緯度経度データ読込み　※左側の数字は行番号です

P.15 基礎知識編３． １０参照

緯度経度データを読込みます。
緯度経度データは AM2_COMMON_LATLON 構造体の 2 次元配列で、サイズはピクセル数×スキャン数です。
緯度経度データの読込みには、AMTK_getLatLon()関数を使用します。

ret=AMTK_getLatLon(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
166   // read array: latlon
167   vpnt=ll;
168   ret=AMTK_getLatLon(hnd,(AM2_COMMON_LATLON **)&vpnt,1,num
,AM2_LATLON_L2_LO);
169   if(ret<0){
170     printf("AMTK_getLatLon error: AM2_LATLON_L2_LO¥n");
171     printf("amtk status: %d¥n",ret);
172     exit(1);
173   }
174   printf("latlon[scan=0][pixel=0]: (%9.4f,%9.4f)¥n", ll[0][0].lat,
ll[0][0].lon);
```

◇物理量データ読込み　※左側の数字は行番号です

物理量データを読込みます。
物理量データは float 型の 3 次元配列で、サイズはレイヤ数×ピクセル数×スキャン数です。
出力が float 型なので、AMTK_get_SwathFloat()関数を使用します。
積雪深(SND)・海面水温(SST)は物理量が 2 層あるので、この 2 つとそれ以外で処理を分けています。
積雪深の 2 層目は積雪水量[cm]です。海面水温の 2 層目は 10GHz による SST[℃]です。

ret=AMTK_get_SwathFloat(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
176   // read array: geophysical data for 1 layer
177   if(strncmp(gid+29,"SND",3)!=0 && strncmp(gid+29,"SST",3)!=0 ){
178     vpnt=geo1;
179     ret=AMTK_get_SwathFloat(hnd,(float **)&vpnt,1,num,AM2_SWATH_GEO1);
180     if(ret<0){
181       printf("AMTK_get_SwathFloat error: AM2_SWATH_GEO1¥n");
182       printf("amtk status: %d¥n",ret);
183       exit(1);
184     }
185   }
186
```

2 層ある場合は、アクセスラベルを変更して読み分けます。

```
187   // read array: geophysical data for 2 layer
188   if(strncmp(gid+29,"SND",3)==0 || strncmp(gid+29,"SST",3)==0 ){
189     // layer 1
190     vpnt=geo1;
191     ret=AMTK_get_SwathFloat(hnd,(float **)&vpnt,1,num,AM2_SWATH_GEO1);
192     if(ret<0){
193       printf("AMTK_get_SwathFloat error: AM2_SWATH_GEO1¥n");
194       printf("amtk status: %d¥n",ret);
195       exit(1);
196     }
197     // layer 2
198     vpnt=geo2;
199     ret=AMTK_get_SwathFloat(hnd,(float **)&vpnt,1,num,AM2_SWATH_GEO2);
200     if(ret<0){
201       printf("AMTK_get_SwathFloat error: AM2_SWATH_GEO2¥n");
202       printf("amtk status: %d¥n",ret);
203       exit(1);
204     }
205   }
```

◇L2 品質フラグデータ読込み　※左側の数字は行番号です

L2 品質フラグデータを読込みます。
L2 品質フラグデータは unsigned char 型の 3 次元配列で、
サイズはピクセル数×スキャン数×レイヤ数です。
出力が unsigned char 型なので、AMTK_get_SwathUChar()関数を使用します。
積雪深(SND)、海面水温(SST)は品質フラグが 2 層あるのでこの 2 つとそれ以外で処理を分けています。

ret=AMTK_get_SwathUChar(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
207   // read array: pixel data quality for 1 layer
208   if(strncmp(gid+29,"SND",3)!=0 && strncmp(gid+29,"SST",3)!=0 ){
209     vpnt=pdq1;
210     ret=AMTK_get_SwathUChar(hnd,(unsigned char **)&vpnt,1,num,AM2_PIX_QUAL);
211     if(ret<0){
212       printf("AMTK_get_SwathUChar error: AM2_PIX_QUAL¥n");
213       printf("amtk status: %d¥n",ret);
214       exit(1);
215     }
216   }
217
```

2 層ある場合は、一時変数で全体を読込んだ後に、層ごとに分割しています。

```
218   // read array: pixel data quality for 2 layer
219   if(strncmp(gid+29,"SND",3)==0 || strncmp(gid+29,"SST",3)==0 ){
220     // read
221     vpnt=pdqtmp;
222     ret=AMTK_get_SwathUChar(hnd,(unsigned char **)&vpnt,1,num,AM2_PIX_QUAL);
223     if(ret<0){
224       printf("AMTK_get_SwathUChar error: AM2_PIX_QUAL¥n");
225       printf("amtk status: %d¥n",ret);
226       exit(1);
227     }
228     // separate
229     for(j=0;j<num;++j){
230       for(i=0;i<AM2_DEF_SNUM_LO;++i){
231         pdq1[j][i]=pdqtmp[num*0+j][i];
232         pdq2[j][i]=pdqtmp[num*1+j][i];
233       }
234     }
235   }
```

L2 品質フラグには、アルゴリズム開発者が設定した物理量算出に係る補足情報が格納されています。
0～15 は OK 状態、16～255 は NG 状態を現します。
NG 状態の場合、物理量には欠損値(-32768)または異常値(-32767)が格納されています。
L2 品質フラグの詳細については、「AMSR2 高次プロダクトフォーマット説明書」(*1)を参照ください。
(*1)http://suzaku.eorc.jaxa.jp/GCOM_W/data/data_w_format_j.html

◇終了処理　※左側の数字は行番号です

ファイルをクローズします。

ret=AMTK_closeH5(hnd)
hnd: クローズするファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

```
261   // close
262   ret=AMTK_closeH5(hnd);
```

５．３．２ コンパイル方法(build_readL2L_amtk_c.sh 解説)

コンパイルに使用するスクリプト build_readL2L_amtk_c.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～9 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 AMTK=/home/user1/util/AMTK_AMSR2_1.11
 8 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 9 SZIP=/home/user1/util/szip_2.1
10
```

12 行目には使用するコンパイラを指定します。
インテルコンパイラ(icc)または PG コンパイラ(pgcc)または GNU コンパイラ(gcc)を指定します。

```
11 # compiler
12 cc=icc
13
14 # source filename
15 csrc=readL2L_amtk.c
16
17 # output filename
18 out=readL2L_amtk_c
19
20 # library order
21 lib="-lAMSR2 -lhdf5 -lsz -lz -lm"
22
23 # compile
24 cmd="$cc -g $csrc -o $out -I$AMTK/include -I$HDF5/include -I$SZIP/include
-L$AMTK/lib -L$HDF5/lib -L$SZIP/lib $lib"
25 echo $cmd
26 $cmd
27
28 # garbage
29 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL2L_amtk_c.sh
icc -g readL2L_amtk.c -o readL2L_amtk_c
-I/home/user1/util/AMTK_AMSR2_1.11/include
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/AMTK_AMSR2_1.11/lib
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lAMSR2 -lhdf5 -lsz -lz -lm
```

５．３．３ プログラム実行結果のサンプル

サンプルプログラムでは固定配列を多数使用しているため、環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞
```
$ unlimit
```

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、4 つとも実行してください。
```
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited
```

```
$ ./readL2L_amtk_c GW1AM2_201303011809_125D_L2SGCLWLA0000000.h5
input file: GW1AM2_201303011809_125D_L2SGCLWLA0000000.h5
GeophysicalName: Cloud Liquid Water
GranuleID: GW1AM2_201303011809_125D_L2SGCLWLA0000000
ObservationStartDateTime: 2013-03-01T18:09:10.122Z
EquatorCrossingDateTime: 2013-03-01T18:35:54.849Z
ObservationEndDateTime: 2013-03-01T18:58:26.342Z
NumberOfScans: 1972
limit of NumberOfScans = 2200
OverlapScans: 0
time[scan=0]: 2013/03/01 18:09:10
latlon[scan=0][pixel=0]: (  84.4574, -78.1076)
geo1[scan=0][pixel=0]: -32767.000 [Kg/m2] (PDQ:112)
```

５．４ L2 高解像度データ読込み

降水量(PRC)が L2 高解像度です。
積算雲水量(CLW)・海氷密接度(SIC)・土壌水分量(SMC)・積雪深(SND)・海面水温(SST)・海上風速(SSW)・可降水量(TPW)は L2 低解像度を参照ください。

５．４．１ サンプルプログラム readL2H_amtk.c 解説

サンプルプログラム readL2H_amtk.c では、L2 高解像度データファイルから、以下のメタデータと格納データを読込んで、内容をテキスト表示します。

メタデータ	格納データ
* GeophysicalName - GranuleID - ObservationStartDateTime - EquatorCrossingDateTime - ObservationEndDateTime * NumberOfScans - OverlapScans	* Scan Time * Latitude of Observation Point for 89A - Latitude of Observation Point for 89B * Longitude of Observation Point for 89A - Longitude of Observation Point for 89B * Geophysical Data for 89A - Geophysical Data for 89B * Pixel Data Quality for 89A - Pixel Data Quality for 89B P.15 基礎知識編３．１０参照

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の AMTK 関数は、最初に使用する際に使用説明を記載します。

メタデータ読込みには、AMTK_getMetaDataName()関数を使用します。
配列データ読込みには、出力されるデータ型によって以下の 4 つの関数を使い分けます。

- 出力が時刻構造体の場合 → AMTK_getScanTime()関数
- 出力が緯度経度構造体の場合 → AMTK_getLatLon()関数
- 出力が float 型の場合 → AMTK_get_SwathFloat()関数
- 出力が unsigned char 型の場合 → AMTK_get_SwathUChar()関数

読込むデータ種類の指定は、アクセスラベルで行います。アクセスラベルとは、データ種類を指定するために AMTK で定義されている定数名です。

◇変数宣言　※左側の数字は行番号です

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <string.h>
 4 #include "AMTK.h"
 5
 6 // fixed value
 7 #define LMT 2200 // limit of NumberOfScans
 :
10 int main(int argc, char *argv[]){
11   // interface variable
12   int i,j;       // loop variable
13   int ret;       // return status
14   char buf[512]; // text buffer
15   void *vpnt;    // pointer to void
16   char *fn;      // filename
17   hid_t hnd;     // file handle
18
19   // meta data
20   char geo[512];  // GeophysicalName
21   char gid[512];  // GranuleID
22   char tm1[512];  // ObservationStartDateTime
23   char tm2[512];  // EquatorCrossingDateTime
24   char tm3[512];  // ObservationEndDateTime
25   int num;       // NumberOfScans
26   int ovr;       // OverlapScans
28   // array data
29   AM2_COMMON_SCANTIME st[LMT]; // scantime
30   AM2_COMMON_LATLON ll89a[LMT][AM2_DEF_SNUM_HI];
31   AM2_COMMON_LATLON ll89b[LMT][AM2_DEF_SNUM_HI];
32   float geo1_89a[LMT][AM2_DEF_SNUM_HI];
33   float geo1_89b[LMT][AM2_DEF_SNUM_HI];
34   unsigned char pdq1_89a[LMT][AM2_DEF_SNUM_HI];
35   unsigned char pdq1_89b[LMT][AM2_DEF_SNUM_HI];
```

C 言語用 AMTK ヘッダーをインクルードします。

標準プロダクトのスキャン数上限は 2200 で充分ですが、
準リアルプロダクトを使用する場合は、
2 周回程度繋がる場合があるので、
LMT=9000 としてください。(標準の 4 パス分程度)

AMTK 用インターフェイス変数を宣言します。

物理量名
グラニュール ID
観測開始時刻
赤道通過時刻
観測終了時刻
スキャン数
オーバーラップスキャン数

メタデータ用変数
を宣言します。

時刻データには AM2_COMMON_SCANTIME 構造体を使います。
緯度経度データには AM2_COMMON_LATLON 構造体を使います。
配列の次元数はデータによって異なります。
スキャン数はパスによって若干上下するので、上限を決めて宣言しておきます。
ここでは上限として LMT=2200 を設定しています。
AM2_DEF_SNUM_HI は AMTK に定義されている定数で、高解像度ピクセル数(486)です。
AM2_DEF_SNUM_LO は AMTK に定義されている定数で、低解像度ピクセル数(243)です。

時刻

緯度経度

物理量

品質フラグ

配列データ用変数
を宣言します。

◇開始処理　※左側の数字は行番号です

ファイルをオープンします。

hnd=AMTK_openH5(fn)
fn: オープンするファイル名を指定します
hnd: [戻り値]成功の場合はファイルハンドル値、失敗の場合は負の値

```
47   // open
48   hnd=AMTK_openH5(fn);
49   if(hnd<0){
50     printf("AMTK_openH5 error: %s¥n", fn);
51     printf("amtk status: %d¥n", hnd);
52     exit(1);
53   }
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込みます。

ret=AMTK_getMetaDataName(hnd,met,out)
hnd: ファイルハンドル値を指定します
met: メタデータ名称を指定します
out: メタデータ内容が返されます
ret: [戻り値]成功の場合は読込んだメタデータの文字数、失敗の場合は負の値

AMTK では出力変数はすべてポインタへのポインタとして渡します。
まずポインタを作ってから、そのポインタを渡します。

```
55   // read meta: GeophysicalName
56   vpnt=geo;
57   ret=AMTK_getMetaDataName(hnd,"GeophysicalName",(char **)&vpnt);
58   if(ret<0){
59     printf("AMTK_getMetaDataName error: GeophysicalName¥n");
60     printf("amtk status: %d¥n",ret);
61     exit(1);
62   }
63   printf("GeophysicalName: %s¥n",geo);
```

コンパイラからの警告を避けるために、
void ポインタを適切な型にキャストします。

メタデータからスキャン数を読込みます。
配列データ読込みではスキャン数の指定が必要になるので、ここで読込んでおきます。

```
111  // read meta: NumberOfScans
112  vpnt=buf;
113  ret=AMTK_getMetaDataName(hnd,"NumberOfScans",(char **)&vpnt);
114  if(ret<0){
115    printf("AMTK_getMetaDataName error: NumberOfScans¥n");
116    printf("amtk status: %d¥n",ret);
117    exit(1);
118  }
119  num=atoi(buf);
120  printf("NumberOfScans: %d¥n",num);
```

メタデータは全て文字として取得されるので、
数値に変換しておきます。

◇時刻データ読込み　※左側の数字は行番号です

P.11 基礎知識編３．６参照

時刻データを読込みます。
時刻データは AM2_COMMON_SCANTIME 構造体の 1 次元配列で、サイズはスキャン数です。
時刻データの読込みには、AMTK_getScanTime()関数を使用します。

ret=AMTK_getScanTime(hnd,bgn,end,out)
hnd: ファイルハンドル値を指定します
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
out: 出力データが返されます
ret: [戻り値]失敗の場合は負の値

```
142   // read array: scantime
143   vpnt=st;
144   ret=AMTK_getScanTime(hnd,1,num,(AM2_COMMON_SCANTIME **)&vpnt);
145   if(ret<0){
146     printf("AMTK_getScanTime error.¥n");
147     printf("amtk status: %d¥n",ret);
148     exit(1);
149   }
150   printf("time[scan=0]: %04d/%02d/%02d %02d:%02d:%02d¥n"
151   , st[0].year
152   , st[0].month
153   , st[0].day
154   , st[0].hour
155   , st[0].minute
156   , st[0].second
157   );
```

◇緯度経度データ読込み　※左側の数字は行番号です

P.15 基礎知識編３．１０参照

緯度経度データを読込みます。
緯度経度データは AM2_COMMON_LATLON 構造体の 2 次元配列で、サイズはピクセル数×スキャン数です。
緯度経度データの読込みには、AMTK_getLatLon()関数を使用します。

ret=AMTK_getLatLon(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
159   // read array: latlon for 89a
160   vpnt=ll89a;
161   ret=AMTK_getLatLon(hnd,(AM2_COMMON_LATLON **)&vpnt,1,num
,AM2_LATLON_L2_89A);
162   if(ret<0){
163     printf("AMTK_getLatLon error: AM2_LATLON_L2_89A¥n");
164     printf("amtk status: %d¥n",ret);
165     exit(1);
166   }
167   printf("latlon89a[scan=0][pixel=0]: (%9.4f,%9.4f)¥n", ll89a[0][0].lat,
ll89a[0][0].lon);
```

◇物理量データ読込み　※左側の数字は行番号です

物理量データを読込みます。
物理量データは float 型の 3 次元配列で、サイズはレイヤ数×ピクセル数×スキャン数です。
出力が float 型なので、AMTK_get_SwathFloat()関数を使用します。

ret=AMTK_get_SwathFloat(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
179   // read array: geophysical data for 1 layer for 89a
180   vpnt=geo1_89a;
181   ret=AMTK_get_SwathFloat(hnd,(float **)&vpnt,1,num,AM2_SWATHA_GEO1);
182   if(ret<0){
183     printf("AMTK_get_SwathFloat error: AM2_SWATHA_GEO1¥n");
184     printf("amtk status: %d¥n",ret);
185     exit(1);
186   }
```

◇L2 品質フラグデータ読込み　※左側の数字は行番号です

L2 品質フラグデータを読込みます。
L2 品質フラグデータは unsigned char 型の 3 次元配列で、
サイズはピクセル数×スキャン数×レイヤ数です。
出力が unsigned char 型なので、AMTK_get_SwathUChar()関数を使用します。

ret=AMTK_get_SwathUChar(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
197   // read array: pixel data quality for 1 layer for 89a
198   vpnt=pdq1_89a;
199   ret=AMTK_get_SwathUChar(hnd,(unsigned char **)&vpnt,1,num,AM2_PIX_QUAL_A);
200   if(ret<0){
201     printf("AMTK_get_SwathUChar error: AM2_PIX_QUAL_A¥n");
202     printf("amtk status: %d¥n",ret);
203     exit(1);
204   }
```

L2 品質フラグには、アルゴリズム開発者が設定した物理量算出に係る補足情報が格納されています。
0～15 は OK 状態、16～255 は NG 状態を現します。
NG 状態の場合、物理量には欠損値(-32768)または異常値(-32767)が格納されています。
L2 品質フラグの詳細については、「AMSR2 高次プロダクトフォーマット説明書」(*1)を参照ください。
(*1)http://suzaku.eorc.jaxa.jp/GCOM_W/data/data_w_format_j.html

◇終了処理　※左側の数字は行番号です

ファイルをクローズします。

ret=AMTK_closeH5(hnd)
hnd: クローズするファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

```
261   // close
262   ret=AMTK_closeH5(hnd);
```

５．４．２ コンパイル方法(build_readL2H_amtk_c.sh 解説)

コンパイルに使用するスクリプト build_readL2H_amtk_c.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～9 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 AMTK=/home/user1/util/AMTK_AMSR2_1.11
 8 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 9 SZIP=/home/user1/util/szip_2.1
10
```

12 行目には使用するコンパイラを指定します。
インテルコンパイラ(icc)または PG コンパイラ(pgcc)または GNU コンパイラ(gcc)を指定します。

```
11 # compiler
12 cc=icc
13
14 # source filename
15 csrc=readL2H_amtk.c
16
17 # output filename
18 out=readL2H_amtk_c
19
20 # library order
21 lib="-lAMSR2 -lhdf5 -lsz -lz -lm"
22
23 # compile
24 cmd="$cc -g $csrc -o $out -I$AMTK/include -I$HDF5/include -I$SZIP/include
-L$AMTK/lib -L$HDF5/lib -L$SZIP/lib $lib"
25 echo $cmd
26 $cmd
27
28 # garbage
29 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL2H_amtk_c.sh
icc -g readL2H_amtk.c -o readL2H_amtk_c
-I/home/user1/util/AMTK_AMSR2_1.11/include
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/AMTK_AMSR2_1.11/lib
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lAMSR2 -lhdf5 -lsz -lz -lm
```

５．４．３ プログラム実行結果のサンプル

サンプルプログラムでは固定配列を多数使用しているため、環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞
```
$ unlimit
```

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、4 つとも実行してください。
```
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited
```

```
$ ./readL2H_amtk_c GW1AM2_201303011809_125D_L2SGPRCHA0000000.h5
input file: GW1AM2_201303011809_125D_L2SGPRCHA0000000.h5
GeophysicalName: Precipitation
GranuleID: GW1AM2_201303011809_125D_L2SGPRCHA0000000
ObservationStartDateTime: 2013-03-01T18:09:10.122Z
EquatorCrossingDateTime: 2013-03-01T18:35:54.849Z
ObservationEndDateTime: 2013-03-01T18:58:26.342Z
NumberOfScans: 1972
limit of NumberOfScans = 2200
OverlapScans: 0
time[scan=0]: 2013/03/01 18:09:10
latlon89a[scan=0][pixel=0]: (  84.4188, -77.9502)
latlon89b[scan=0][pixel=0]: (  84.3305, -78.8925)
geo1_89a[scan=0][pixel=0]: -32767.0 [mm/h] (PDQ: 16)
geo1_89b[scan=0][pixel=0]: -32767.0 [mm/h] (PDQ: 16)
```

５．５ L3 輝度温度データ読込み

５．５．１ サンプルプログラム readL3B_amtk.c 解説

サンプルプログラム readL3B_amtk.c では、L3 輝度温度データファイルから、以下のメタデータと格納データを読込んで、内容をテキスト表示します。

メタデータ	格納データ
* GeophysicalName - GranuleID	* Brightness Temperature (H) - Brightness Temperature (V)

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の AMTK 関数は、最初に使用する際に使用説明を記載します。

メタデータ読込みには、AMTK_getMetaDataName()関数を使用します。
配列データ読込みには、AMTK_get_GridFloat()関数を使用します。

読込むデータ種類の指定は、アクセスラベルで行います。アクセスラベルとは、データ種類を指定するために AMTK で定義されている定数名です。

◇変数宣言　※左側の数字は行番号です

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <string.h>
 4 #include "AMTK.h"
 :
 9 int main(int argc, char *argv[]){
10   // interface variable
11   int i,j;       // loop variable
12   int ret;       // return status
13   char buf[512]; // text buffer
14   void *vpnt;    // pointer to void
15   char *fn;      // filename
16   hid_t hnd;     // file handle
17   int siz[3];    // array size
18   int x;        // grid size x
19   int y;        // grid size y
 :
27   // meta data
28   char geo[512];  // GeophysicalName
29   char gid[512];  // GranuleID
30
31   // array data
32   float *tbH; // brightness temperature for horizontal
33   float *tbV; // brightness temperature for vertical
```

C 言語用 AMTK ヘッダーをインクルードします。

AMTK 用インターフェイス変数を宣言します。

物理量名
グラニュール ID

メタデータ用変数を宣言します。

輝度温度

配列データ用変数を宣言します。

この時点では、配列データ用のメモリ領域は確保されていません。
後程、配列サイズを調べてから、メモリ領域の確保を行います。

◇開始処理　※左側の数字は行番号です

ファイルをオープンします。

hnd=AMTK_openH5(fn)
fn: オープンするファイル名を指定します
hnd: [戻り値]成功の場合はファイルハンドル値、失敗の場合は負の値

```
45   // open
46   hnd=AMTK_openH5(fn);
47   if(hnd<0){
48     printf("AMTK_openH5 error: %s¥n", fn);
49     printf("amtk status: %d¥n", hnd);
50     exit(1);
51   }
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込みます。

ret=AMTK_getMetaDataName(hnd,met,out)
hnd: ファイルハンドル値を指定します
met: メタデータ名称を指定します
out: メタデータ内容が返されます
ret: [戻り値]成功の場合は読込んだメタデータの文字数、失敗の場合は負の値

AMTK では出力変数はすべてポインタへのポインタとして渡します。
まずポインタを作ってから、そのポインタを渡します。

```
53   // read meta: GeophysicalName
54   vpnt=geo;
55   ret=AMTK_getMetaDataName(hnd,"GeophysicalName",(char **)&vpnt);
56   if(ret<0){
57     printf("AMTK_getMetaDataName error: GeophysicalName¥n");
58     printf("amtk status: %d¥n",ret);
59     exit(1);
60   }
61   printf("GeophysicalName: %s¥n",geo);
```

コンパイラからの警告を避けるために、
void ポインタを適切な型にキャストします。

◇配列サイズ取得とメモリ領域確保　※左側の数字は行番号です

配列サイズを取得します。
配列サイズの取得には、AMTK_getDimSize()関数を使用します。

ret=AMTK_getDimSize(hnd,label,siz)
hnd: ファイルハンドル値を指定します
label: アクセスラベルを指定します
siz: [戻り値]配列サイズ
ret: [戻り値]失敗の場合は負の値

```
86   // get grid size
87   ret=AMTK_getDimSize(hnd,AM2_GRID_TBH,siz);
88   if(ret<0){
89     printf("AMTK_getDimSize error: AM2_GRID_TBH¥n");
90     printf("amtk status: %d¥n",ret);
91     exit(1);
92   }
93   x=siz[1];
94   y=siz[0];
95   printf("grid size x: %d¥n", x);
96   printf("grid size y: %d¥n", y);
```

メモリ領域を確保します。

```
98   // memory allocate
99   tbH=malloc(sizeof(float)*x*y);
100  if(tbH==NULL){
101    printf("memory allocate error: tbH¥n");
102    exit(1);
103  }
```

◇輝度温度データ読込み　※左側の数字は行番号です

輝度温度データを読込みます。

ret=AMTK_get_GridFloat(hnd,out,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
110   // read horizontal
111   vpnt=tbH;
112   ret=AMTK_get_GridFloat(hnd,(float **)&vpnt,AM2_GRID_TBH);
113   if(ret<0){
114     printf("AMTK_get_GridFloat error: AM2_GRID_TBH¥n");
115     printf("amtk status: %d¥n",ret);
116     exit(1);
117   }
```

◇終了処理　※左側の数字は行番号です

メモリを開放します。

```
212   // memory free
213   free(tbH);
214   free(tbV);
```

ファイルをクローズします。

ret=AMTK_closeH5(hnd)
hnd: クローズするファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

```
216   // close
217   ret=AMTK_closeH5(hnd);
```

５．５．２ コンパイル方法(build_readL3B_amtk_c.sh 解説)

コンパイルに使用するスクリプト build_readL3B_amtk_c.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～9 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 AMTK=/home/user1/util/AMTK_AMSR2_1.11
 8 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 9 SZIP=/home/user1/util/szip_2.1
10
```

12 行目には使用するコンパイラを指定します。
インテルコンパイラ(icc)または PG コンパイラ(pgcc)または GNU コンパイラ(gcc)を指定します。

```
11 # compiler
12 cc=icc
13
14 # source filename
15 csrc=readL3B_amtk.c
16
17 # output filename
18 out=readL3B_amtk_c
19
20 # library order
21 lib="-lAMSR2 -lhdf5 -lsz -lz -lm"
22
23 # compile
24 cmd="$cc -g $csrc -o $out -I$AMTK/include -I$HDF5/include -I$SZIP/include
-L$AMTK/lib -L$HDF5/lib -L$SZIP/lib $lib"
25 echo $cmd
26 $cmd
27
28 # garbage
29 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL3B_amtk_c.sh
icc -g readL3B_amtk.c -o readL3B_amtk_c
-I/home/user1/util/AMTK_AMSR2_1.11/include
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/AMTK_AMSR2_1.11/lib
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lAMSR2 -lhdf5 -lsz -lz -lm
```

５．５．３ プログラム実行結果のサンプル

環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞
$ unlimit

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、
4 つとも実行してください。
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited

```
$ ./readL3B_amtk_c GW1AM2_20130200_01M_EQMA_L3SGT06LA1110110.h5
input file: GW1AM2_20130200_01M_EQMA_L3SGT06LA1110110.h5
GeophysicalName: Brightness Temperature (6GHz)
GranuleID: GW1AM2_20130200_01M_EQMA_L3SGT06LA1110110
grid size x: 1440
grid size y: 720

ASCII ART OF HORIZONTAL BRIGHTNESS TEMPERATURE (X/20GRID Y/40GRID)
+------------------------------------------------------------------------+
|                                                                        |
|4112333444434444444444444444444444444444444444444444444444433443222222444444|
|1111343111134444444444444444444444444444444444444444444444444434233343111|
|1245234455555555444444444444444434312343444144555444444444442112111111111|
|1545444444444444444444444445441111111111111111111355444444444222111111111112|
|4122425544144445444444453411311111111111111111111455544445511111111111115|
|4444444442444555445555441111111111111111111111155555351111111111111124|
|4444444424441115551555311111111111111111121111111155151141111111111444|
|5555555555111112111131112111111111111111111111111111141554111111111155|
|1155555551111111111515531111111111111111111111111111114555554111111111|
|1115555531111111111111114211113121111111111111111111111555555551111111|
|1115555215111111111111114555551111111111111111111111111145555521111111|
|1111555111111111111111135555555111111111111111111111111154552111111111|
|1111111111111111111111111111111113111111111111111111111155111111111111|
|1111111111111111111111111111111111111111111111111111111351111111111111|
|1111111111111111111111111111111111111111111111111111111111111111111111|
|3433234443333424343343444444444334332111111111111111111134444444111133|
|3333333333333333333333333333333343333333222223333333443344333333333333|
+------------------------------------------------------------------------+
```

```
ASCII ART OF VERTICAL BRIGHTNESS TEMPERATURE (X/20GRID Y/40GRID)
+------------------------------------------------------------------------+
|                                                                        |
|43334444444445545544444444444444444444445444444444444444444433333244545|
|33334543333455455444454444444444444444444454544444444444444455443443443333|
|33553445555555555555455544545555444334545543455555444455444433333333333|
|35555555555555555545555555555433333333333333333345555555555433333333333333|
|43334355553555555555555545334333333333333333333355555555553333333333333335|
|55555555554555555555555553333333333333333333333355555453333333333333335|
|55555555355533355535554333333333333333333333333333553533433333333333555|
|65556665555333333333433343333333333333333333333333333535555333333333355|
|33555555633333333333535543333333333333333333333333333335555554333333333|
|33355555433333333333333433333433333333333333333333333335555555633333333|
|33355553353333333333333566665333333333333333333333333335555554333333333|
|33336553333333333333334666666533333333333333333333333335555433333333333|
|33333333333333333333333333333333333433333333333333333335533333333333333|
|33333333333333333333333333333333333333333333333333333345333333333333333|
|33333333333333333333333333333333333333333333333333333333333333333333333|
|44433444444444344444444444444444444433333333333333333334445555433333344|
|44444443344444444443333443444444444444444433444444444444444444444444444|
+------------------------------------------------------------------------+

[#]:missing
[ ]:out of observation
[1]: 50-100K
[2]:100-150K
[3]:150-200K
[4]:200-250K
[5]:250-300K
[6]:300-350K
[*]:other
```

５．６ L3 物理量データ読込み

５．６．１ サンプルプログラム readL3G_amtk.c 解説

サンプルプログラム readL3G_amtk.c では、L3 物理量データファイルから、以下のメタデータと格納データを読込んで、内容をテキスト表示します。

メタデータ	格納データ
* GeophysicalName - GranuleID	* Geophysical Data

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の AMTK 関数は、最初に使用する際に使用説明を記載します。

メタデータ読込みには、AMTK_getMetaDataName()関数を使用します。
配列データ読込みには、AMTK_get_GridFloat()関数を使用します。

読込むデータ種類の指定は、アクセスラベルで行います。アクセスラベルとは、データ種類を指定するために AMTK で定義されている定数名です。

◇変数宣言　※左側の数字は行番号です

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <string.h>
 4 #include "AMTK.h"
 :
 9 int main(int argc, char *argv[]){
10   // interface variable
11   int i,j;       // loop variable
12   int ret;       // return status
13   char buf[512]; // text buffer
14   void *vpnt;    // pointer to void
15   char *fn;      // filename
16   hid_t hnd;     // file handle
17   int siz[3];    // array size
18   int x;        // grid size x
19   int y;        // grid size y
 :
29   // meta data
30   char geo[512];  // GeophysicalName
31   char gid[512];  // GranuleID
32
33   // array data
34   float *geo1; // geophysical data layer 1
35   float *geo2; // geophysical data layer 2
```

C 言語用 AMTK ヘッダーをインクルードします。

AMTK 用インターフェイス変数を宣言します。

物理量名
グラニュール ID

メタデータ用変数を宣言します。

物理量

配列データ用変数を宣言します。

この時点では、配列データ用のメモリ領域は確保されていません。
後程、配列サイズを調べてから、メモリ領域の確保を行います。

◇開始処理　※左側の数字は行番号です

ファイルをオープンします。

hnd=AMTK_openH5(fn)
fn: オープンするファイル名を指定します
hnd: [戻り値]成功の場合はファイルハンドル値、失敗の場合は負の値

```
47   // open
48   hnd=AMTK_openH5(fn);
49   if(hnd<0){
50     printf("AMTK_openH5 error: %s¥n", fn);
51     printf("amtk status: %d¥n", hnd);
52     exit(1);
53   }
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込みます。

ret=AMTK_getMetaDataName(hnd,met,out)
hnd: ファイルハンドル値を指定します
met: メタデータ名称を指定します
out: メタデータ内容が返されます
ret: [戻り値]成功の場合は読込んだメタデータの文字数、失敗の場合は負の値

AMTK では出力変数はすべてポインタへのポインタとして渡します。
まずポインタを作ってから、そのポインタを渡します。

```
55   // read meta: GeophysicalName
56   vpnt=geo;
57   ret=AMTK_getMetaDataName(hnd,"GeophysicalName",(char **)&vpnt);
58   if(ret<0){
59     printf("AMTK_getMetaDataName error: GeophysicalName¥n");
60     printf("amtk status: %d¥n",ret);
61     exit(1);
62   }
63   printf("GeophysicalName: %s¥n",geo);
```

コンパイラからの警告を避けるために、
void ポインタを適切な型にキャストします。

◇配列サイズ取得とメモリ領域確保　※左側の数字は行番号です

配列サイズを取得します。
配列サイズの取得には、AMTK_getDimSize()関数を使用します。

ret=AMTK_getDimSize(hnd,label,siz)
hnd: ファイルハンドル値を指定します
label: アクセスラベルを指定します
siz: [戻り値]配列サイズ
ret: [戻り値]失敗の場合は負の値

```
89   // get grid size
90   ret=AMTK_getDimSize(hnd,AM2_GRID_GEO1,siz);
91   if(ret<0){
92     printf("AMTK_getDimSize error: AM2_GRID_GEO1¥n");
93     printf("amtk status: %d¥n",ret);
94     exit(1);
95   }
96   x=siz[1];
97   y=siz[0];
98   printf("grid size x: %d¥n", x);
99   printf("grid size y: %d¥n", y);
```

メモリ領域を確保します。

```
101  // memory allocate layer 1
102  geo1=malloc(sizeof(float)*x*y);
103  if(geo1==NULL){
104    printf("memory allocate error: geo1¥n");
105    exit(1);
106  }
```

◇物理量データ読込み　※左側の数字は行番号です

物理量データを読込みます。
積雪深(SND)・海面水温(SST)は物理量が 2 層あるので、この 2 つとそれ以外で処理を分けています。
積雪深の 2 層目は積雪水量[cm]です。海面水温の 2 層目は 10GHz による SST[℃]です。

ret=AMTK_get_GridFloat(hnd,out,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
117   // read layer 1
118   vpnt=geo1;
119   ret=AMTK_get_GridFloat(hnd,(float **)&vpnt,AM2_GRID_GEO1);
120   if(ret<0){
121     printf("AMTK_get_GridFloat error: AM2_GRID_GEO1¥n");
122     printf("amtk status: %d¥n",ret);
123     exit(1);
124   }
125
```

2 層ある場合は、アクセスラベルを変更して読み分けます。

```
126   // read layer 2
127   if(strncmp(gid+29,"SND",3)==0 || strncmp(gid+29,"SST",3)==0 ){
128     vpnt=geo2;
129     ret=AMTK_get_GridFloat(hnd,(float **)&vpnt,AM2_GRID_GEO2);
130     if(ret<0){
131       printf("AMTK_get_GridFloat error: AM2_GRID_GEO2¥n");
132       printf("amtk status: %d¥n",ret);
133       exit(1);
134     }
135   }
```

◇終了処理　※左側の数字は行番号です

メモリを開放します。

```
317   // memory free
318   free(geo1);
319   if(strncmp(gid+29,"SND",3)==0 || strncmp(gid+29,"SST",3)==0 ){
320     free(geo2);
321   }
```

ファイルをクローズします。

ret=AMTK_closeH5(hnd)
hnd: クローズするファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

```
323   // close
324   ret=AMTK_closeH5(hnd);
```

５．６．２ コンパイル方法(build_readL3G_amtk_c.sh 解説)

コンパイルに使用するスクリプト build_readL3G_amtk_c.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～9 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 AMTK=/home/user1/util/AMTK_AMSR2_1.11
 8 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 9 SZIP=/home/user1/util/szip_2.1
10
```

12 行目には使用するコンパイラを指定します。
インテルコンパイラ(icc)または PG コンパイラ(pgcc)または GNU コンパイラ(gcc)を指定します。

```
11 # compiler
12 cc=icc
13
14 # source filename
15 csrc=readL3G_amtk.c
16
17 # output filename
18 out=readL3G_amtk_c
19
20 # library order
21 lib="-lAMSR2 -lhdf5 -lsz -lz -lm"
22
23 # compile
24 cmd="$cc -g $csrc -o $out -I$AMTK/include -I$HDF5/include -I$SZIP/include
-L$AMTK/lib -L$HDF5/lib -L$SZIP/lib $lib"
25 echo $cmd
26 $cmd
27
28 # garbage
29 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL3G_amtk_c.sh
icc -g readL3G_amtk.c -o readL3G_amtk_c
-I/home/user1/util/AMTK_AMSR2_1.11/include
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/AMTK_AMSR2_1.11/lib
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lAMSR2 -lhdf5 -lsz -lz -lm
```

５．６．３ プログラム実行結果のサンプル

環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞
$ unlimit

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、
4 つとも実行してください。
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited

```
$ ./readL3G_amtk_c GW1AM2_20130200_01M_EQMA_L3SGCLWLA0000000.h5
input file: GW1AM2_20130200_01M_EQMA_L3SGCLWLA0000000.h5
GeophysicalName: Cloud Liquid Water
GranuleID: GW1AM2_20130200_01M_EQMA_L3SGCLWLA0000000
grid size x: 1440
grid size y: 720

ASCII ART OF GEOPHYSICAL DATA LAYER #1 (X/20GRID Y/40GRID)
+------------------------------------------------------------------------+
|                                                                        |
|#21#####################################################################|
|2212###1113#############################################################222|
|1#################################11######3################011#42223322|
|2##############################00101112413333233##############343413131#|
|#20#######2###############00#122345234421321112O#########3355421131121#|
|########################4#32222232321001101111O#######14*41111201111##|
|########0###000###0####1111111121012011O*131121100##1#11#01211110100###|
|###########1011111111#211#11111210011121333223224321000#1###1111000000##|
|11#######00010122125#3###1#122323211000100201100010111122#######222321111|
|000######122131112311112#01112#113122123231101000000000110########1001110|
|100####32#12111141111000######223232242121244133111001101O######11111100|
|1000###121123221125110O########131112232110121313122011010#####231111001|
|11131212231110111111211111110012102#123122122331121311102l##011124223242|
|*33222511232222222223223331421022422322211024222222122113##0111421221132|
|10112211222111122211111111112223123223233124332121232322122331223222211l|
|#####################################3111221222211121122l########21221##|
|#####################################################################|
+------------------------------------------------------------------------+

[#]:missing
[ ]:out of observation
[0]:  0.000 -  0.030 Kg/m2
[1]:  0.030 -  0.060 Kg/m2
[2]:  0.060 -  0.090 Kg/m2
[3]:  0.090 -  0.120 Kg/m2
[4]:  0.120 -  0.150 Kg/m2
[5]:  0.150 -  0.180 Kg/m2
[*]:other
```

６．AMTK 編(FORTRAN90)

赤文字の解説はサンプルプログラムについて説明しています。

青文字の解説は関数リファレンスまたは衛星基礎知識について説明しています。

６．１ L1B データ読込み

６．１．１ サンプルプログラム readL1B_amtk.f 解説

サンプルプログラム readL1B_amtk.f では、L1B データファイルから、以下のメタデータと格納データを読込んで、89G A ホーン緯度経度から低周波緯度経度を算出し、内容をテキスト表示します。

メタデータ	格納データ
* GeophysicalName - GranuleID - ObservationStartDateTime - EquatorCrossingDateTime - ObservationEndDateTime * NumberOfScans - OverlapScans - CoRegistrationParameterA1 - CoRegistrationParameterA2	* Scan Time * Latitude of Observation Point for 89A - Latitude of Observation Point for 89B - Longitude of Observation Point for 89A - Longitude of Observation Point for 89B * Brightness Temperature (6.9GHz,H) - Brightness Temperature (6.9GHz,V) - Brightness Temperature (7.3GHz,H) - Brightness Temperature (7.3GHz,V) - Brightness Temperature (10.7GHz,H) - Brightness Temperature (10.7GHz,V) - Brightness Temperature (18.7GHz,H) - Brightness Temperature (18.7GHz,V) - Brightness Temperature (23.8GHz,H) - Brightness Temperature (23.8GHz,V) - Brightness Temperature (36.5GHz,H) - Brightness Temperature (36.5GHz,V) - Brightness Temperature (89.0GHz-A,H) - Brightness Temperature (89.0GHz-A,V) - Brightness Temperature (89.0GHz-B,H) - Brightness Temperature (89.0GHz-B,V) * Pixel Data Quality 6 to 36 - Pixel Data Quality 89 * Land_Ocean Flag 6 to 36 - Land_Ocean Flag 89 * Earth Incidence - Earth Azimuth
89G A ホーン緯度経度から算出するデータ - 緯度経度(低周波平均) - 緯度経度(6G) - 緯度経度(7G) - 緯度経度(10G) - 緯度経度(18G) - 緯度経度(23G) - 緯度経度(36G) → P.15 基礎知識編３．１０参照	

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の AMTK 関数は、最初に使用する際に使用説明を記載します。

メタデータ読込みには、AMTK_getMetaDataName()関数を使用します。
配列データ読込みには、出力されるデータ型によって以下の 4 つの関数を使い分けます。

- 出力が時刻構造体の場合 → AMTK_getScanTime()関数
- 出力が緯度経度構造体の場合 → AMTK_getLatLon()関数
- 出力が real(4)型の場合 → AMTK_get_SwathFloat()関数
- 出力が integer(4)型の場合 → AMTK_get_SwathInt()関数

読込むデータ種類の指定は、アクセスラベルで行います。アクセスラベルとは、データ種類を指定するために AMTK で定義されている定数名です。

◇変数宣言　※左側の数字は行番号です

```
 1      program main
 2      implicit none
 3 C include
 4      include 'AMTK_f.h'
 :
 7 C fixed value
 8      integer(4),parameter::LMT=2200 ! limit of NumberOfScans
 9 C interface variable
10      integer(4) i,j          ! loop variable
11      integer(4) ret          ! return status
12      character(len=512) buf ! text buffer
13      character(len=512) fn  ! filename
14      integer(4) hnd          ! file handle
15 C meta data
16      character(len=512) geo  ! GeophysicalName
17      character(len=512) gid  ! GranuleID
18      character(len=512) tm1  ! ObservationStartD
19      character(len=512) tm2  ! EquatorCrossingDa
20      character(len=512) tm3  ! ObservationEndDat
21      integer(4) num          ! NumberOfScans
22      integer(4) ovr          ! OverlapScans
23      character(len=512) prm1 ! CoRegistrationPar
24      character(len=512) prm2 ! CoRegistrationPar
 :
25 C array data
26      type(AM2_COMMON_SCANTIME) st(LMT) ! scantime
27      type(AM2_COMMON_LATLON) ll89a(AM2_DEF_SNUM_HI,LMT) ! latlon for 89a
28      type(AM2_COMMON_LATLON) ll89b(AM2_DEF_SNUM_HI,LMT) ! latlon for 89b
 :
36      real(4) tb06h(AM2_DEF_SNUM_LO,LMT)  ! tb for 06h
37      real(4) tb06v(AM2_DEF_SNUM_LO,LMT)  ! tb for 06v
 :
53      integer(1) pdq06h(AM2_DEF_SNUM_LO,LMT)   ! pixel data quality for 06h
54      integer(1) pdq06v(AM2_DEF_SNUM_LO,LMT)   ! pixel data quality for 06v
 :
63      integer(1) lof06(AM2_DEF_SNUM_LO,LMT)   ! land ocean flag for 06
64      integer(1) lof07(AM2_DEF_SNUM_LO,LMT)   ! land ocean flag for 07
 :
72      real(4) ear_in(AM2_DEF_SNUM_LO,LMT) ! earth incidence
73      real(4) ear_az(AM2_DEF_SNUM_LO,LMT) ! earth azimuth
```

FORTRAN 用 AMTK ヘッダーをインクルードします。

標準プロダクトのスキャン数上限は 2200 で充分ですが、
準リアルプロダクトを使用する場合は、
2 周回程度繋がる場合があるので、
LMT=9000 としてください。(標準の 4 パス分程度)

AMTK 用インターフェイス変数を宣言します。

メタデータ用変数を宣言します。

物理量名
グラニュール ID
観測開始時刻
赤道通過時刻
観測終了時刻
スキャン数
オーバーラップスキャン数
相対レジストレーション係数 A1
相対レジストレーション係数 A2

時刻データには AM2_COMMON_SCANTIME 構造体を使います。
緯度経度データには AM2_COMMON_LATLON 構造体を使います。
配列の次元数はデータによって異なります。
スキャン数はパスによって若干上下するので、上限を決めて宣言しておきます。
ここでは上限として LMT=2200 を設定しています。
AM2_DEF_SNUM_HI は AMTK に定義されている定数で、高解像度ピクセル数(486)です。
AM2_DEF_SNUM_LO は AMTK に定義されている定数で、低解像度ピクセル数(243)です。

時刻
緯度経度
輝度温度
品質フラグ
陸海フラグ
観測入射角
観測方位角

配列データ用変数を宣言します。

◇開始処理　※左側の数字は行番号です

ファイルをオープンします。

hnd=AMTK_openH5(fn)
fn: オープンするファイル名を指定します
hnd: [戻り値]成功の場合はファイルハンドル値、失敗の場合は負の値

```
 82C open
 83      hnd=AMTK_openH5(fn)
 84      if(hnd.lt.0)then
 85        write(*,'(a,a)')'AMTK_openH5 error: ',fn(1:len_trim(fn))
 86        write(*,'(a,i12)')'amtk status: ',hnd
 87        call exit(1)
 88      endif
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込みます。

ret=AMTK_getMetaDataName(hnd,met,out)
hnd: ファイルハンドル値を指定します
met: メタデータ名称を指定します
out: メタデータ内容が返されます
ret: [戻り値]成功の場合は読込んだメタデータの文字数、失敗の場合は負の値

```
 89C read meta: GeophysicalName
 90      ret=AMTK_getMetaDataName(hnd,'GeophysicalName',geo)
 91      if(ret.lt.0)then
92        write(*,'(a)')'AMTK_getMetaDataName error: GeophysicalName'
93        write(*,'(a,i12)')'amtk status: ',ret
94        call exit(1)
95      endif
96      write(*,'(a,a)')'GeophysicalName: ',geo(1:len_trim(geo))
```

メタデータからスキャン数を読込みます。
配列データ読込みではスキャン数の指定が必要になるので、ここで読込んでおきます。

```
132C read meta: NumberOfScans
133      ret=AMTK_getMetaDataName(hnd,'NumberOfScans',buf)
134      if(ret.lt.0)then
135        write(*,'(a)')'AMTK_getMetaDataName error: NumberOfScans'
136        write(*,'(a,i12)')'amtk status: ',ret
137        call exit(1)
138      endif
139      read(buf(1:ret),*)num
140      write(*,'(a,i12)')'NumberOfScans: ',num
```

メタデータは全て文字として取得されるので、数値に変換しておきます。

◇時刻データ読込み　※左側の数字は行番号です

P.11 基礎知識編３． ６参照

時刻データを読込みます。
時刻データは AM2_COMMON_SCANTIME 構造体の 1 次元配列で、サイズはスキャン数です。
時刻データの読込みには、AMTK_getScanTime()関数を使用します。

ret=AMTK_getScanTime(hnd,bgn,end,out)
hnd: ファイルハンドル値を指定します
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
out: 出力データが返されます
ret: [戻り値]失敗の場合は負の値

```
177C read array: scantime
178       ret=AMTK_getScanTime(hnd,1,num,st)
179       if(ret.lt.0)then
180         write(*,'(a)')'AMTK_getScanTime error.'
181         write(*,'(a,i12)')'amtk status: ',ret
182         call exit(1)
183       endif
184       write(*,'(a,i4.4,"/",i2.2,"/",i2.2," ",i2.2,":",i2.2,":",i2.2)')
185      +'time(scan=1): '
186      +,st(1)%year
187      +,st(1)%month
188      +,st(1)%day
189      +,st(1)%hour
190      +,st(1)%minute
191      +,st(1)%second
```

◇緯度経度データ読込み　※左側の数字は行番号です

P.15 基礎知識編３． １０参照

緯度経度データを読込みます。
緯度経度データは AM2_COMMON_LATLON 構造体の 2 次元配列で、サイズはピクセル数×スキャン数です。
緯度経度データの読込みには、AMTK_getLatLon()関数を使用します。

ret=AMTK_getLatLon(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
192 C read array: latlon for 89a
193       ret=AMTK_getLatLon(hnd,ll89a,1,num,AM2_LATLON_89A)
194       if(ret.lt.0)then
195         write(*,'(a)')'AMTK_getLatLon error: AM2_LATLON_89A'
196         write(*,'(a,i12)')'amtk status: ',ret
197         call exit(1)
198       endif
199       write(*,'(a,"(",f9.4,",",f9.4,")")')'latlon89a(pixel=1,scan=1): '
200      +,ll89a(1,1)%lat,ll89a(1,1)%lon
```

◇輝度温度データ読込み　※左側の数字は行番号です

輝度温度データを読込みます。
輝度温度データは real(4)型の 2 次元配列で、サイズはピクセル数×スキャン数です。
出力が real(4)型なので、AMTK_get_SwathFloat()関数を使用します。

ret=AMTK_get_SwathFloat(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
273 C read array: tb for 06h
274       ret=AMTK_get_SwathFloat(hnd,tb06h,1,num,AM2_TB06H)
275       if(ret.lt.0)then
276         write(*,'(a)')'AMTK_get_SwathFloat error: AM2_TB06H'
277         write(*,'(a,i12)')'amtk status: ',ret
278         call exit(1)
279       endif
280       write(*,'(a,f9.2)')'tb06h(pixel=1,scan=1): ',tb06h(1,1)
```

◇L1 品質フラグデータ読込み　※左側の数字は行番号です

L1 品質フラグデータを読込みます。
L1 品質フラグデータは integer(4)型の 2 次元配列で、
L1 低解像度品質フラグデータのサイズは(ピクセル数*16)×スキャン数です。
L1 高解像度品質フラグデータのサイズは(ピクセル数* 8)×スキャン数です。
出力が integer(4)型なので、AMTK_get_SwathInt()関数を使用します。

ret=AMTK_get_SwathInt(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
401 C read array: pixel data quality for low
402       ret=AMTK_get_SwathInt(hnd,pdqlo,1,num,AM2_PIX_QUAL_LO)
403       if(ret.lt.0)then
404         write(*,'(a)')'AMTK_get_SwathInt error: AM2_PIX_QUAL_LO'
405         write(*,'(a,i12)')'amtk status: ',ret
406         call exit(1)
407       endif
```

L1 品質フラグデータは、各周波数（6GHz、7GHz)、偏波に対する RFI 情報がまとまって取得されます。
まとまったままでは扱い難いので、各周波数毎の配列に分割します。
格納値は 2 ビットのマスク値なので、各周波数毎に integer(1)型の 2 次元配列を用意します。
L1 品質フラグデータには RFI(Radio Frequency Interference; 電波干渉)情報が格納されています。
電波干渉の発生がない場合は 00、発生の可能性がある場合は 10、発生がある場合は 11 となります。

```
408       do j=1,num
409         do i=1,AM2_DEF_SNUM_LO
410           pdq06v(i,j)=pdqlo((i-1)*16+ 2,j)*10+pdqlo((i-1)*16+ 1,j)
411           pdq06h(i,j)=pdqlo((i-1)*16+ 4,j)*10+pdqlo((i-1)*16+ 3,j)
412           pdq07v(i,j)=pdqlo((i-1)*16+ 6,j)*10+pdqlo((i-1)*16+ 5,j)
413           pdq07h(i,j)=pdqlo((i-1)*16+ 8,j)*10+pdqlo((i-1)*16+ 7,j)
414         enddo
415       enddo
```

◇L1B 陸海フラグデータ読込み　※左側の数字は行番号です

陸海フラグデータを読込みます。
陸海フラグデータは integer(4)型の 2 次元配列で、
L1B 低解像度陸海データのサイズはピクセル数×(スキャン数*6)です。
L1B 高解像度陸海データのサイズはピクセル数×(スキャン数*2)です。
出力が integer(4)型なので、AMTK_get_SwathInt()関数を使用します。

→ P.14 基礎知識編３． ９参照

```
439 C read array: land ocean flag for low
440       ret=AMTK_get_SwathInt(hnd,loflo,1,num,AM2_LOF_LO)
441       if(ret.lt.0)then
442         write(*,'(a)')'AMTK_get_SwathInt error: AM2_LOF_LO'
443         write(*,'(a,i12)')'amtk status: ',ret
444         call exit(1)
445       endif
```

陸海フラグデータは、解像度毎に全周波数データがまとまって取得されます。
まとまったままでは扱い難いので、各周波数毎の配列に分割します。
格納値は 0～100 のフラグ値なので、各周波数毎に integer(1)型の 2 次元配列を用意します。

```
446       do j=1,num
447         do i=1,AM2_DEF_SNUM_LO
448           lof06(i,j)=loflo(i,num*0+j)
449           lof07(i,j)=loflo(i,num*1+j)
450           lof10(i,j)=loflo(i,num*2+j)
451           lof18(i,j)=loflo(i,num*3+j)
452           lof23(i,j)=loflo(i,num*4+j)
453           lof36(i,j)=loflo(i,num*5+j)
454         enddo
455       enddo
```

◇観測入射角データ読込み　※左側の数字は行番号です

観測入射角データを読込みます。
観測入射角データは real(4)型の 2 次元配列で、サイズはピクセル数×スキャン数です。
出力が real(4)型なので、AMTK_get_SwathFloat()関数を使用します。

```
477 C read array: earth incidence
478       ret=AMTK_get_SwathFloat(hnd,ear_in,1,num,AM2_EARTH_INC)
479       if(ret.lt.0)then
480         write(*,'(a)')'AMTK_get_SwathFloat error: AM2_EARTH_INC'
481         write(*,'(a,i12)')'amtk status: ',ret
482         call exit(1)
483       endif
484       write(*,'(a,f9.2)')'ear_in(pixel=1,scan=1): ',ear_in(1,1)
```

◇終了処理　※左側の数字は行番号です

ファイルをクローズします。

ret=AMTK_closeH5(hnd)
hnd: クローズするファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

```
493 C close
494       ret=AMTK_closeH5(hnd)
```

６．１．２ コンパイル方法(build_readL1B_amtk_f.sh 解説)

コンパイルに使用するスクリプト build_readL1B_amtk_f.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～9 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 AMTK=/home/user1/util/AMTK_AMSR2_1.11
 8 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 9 SZIP=/home/user1/util/szip_2.1
10
```

12 行目には使用するコンパイラを指定します。
インテルコンパイラ(ifort)または PG コンパイラ(pgf90)を指定します。

```
11 # compiler
12 fc=ifort
13
14 # source filename
15 fsrc=readL1B_amtk.f
16
17 # output filename
18 out=readL1B_amtk_f
19
20 # library order
21 lib="-lAMSR2 -lhdf5 -lsz -lz -lm"
22
23 # compile
24 cmd="$fc -g $fsrc -o $out -I$AMTK/include -I$HDF5/include -I$SZIP/include
-L$AMTK/lib -L$HDF5/lib -L$SZIP/lib $lib"
25 echo $cmd
26 $cmd
27
28 # garbage
29 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL1B_amtk_f.sh
ifort -g readL1B_amtk.f -o readL1B_amtk_f
-I/home/user1/util/AMTK_AMSR2_1.11/include
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/AMTK_AMSR2_1.11/lib
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lAMSR2 -lhdf5 -lsz -lz -lm
```

６．１．３ プログラム実行結果のサンプル

サンプルプログラムでは固定配列を多数使用しているため、環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞
```
$ unlimit
```

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、4 つとも実行してください。
```
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited
```

```
$ ./readL1B_amtk_f GW1AM2_201207261145_055A_L1SGBTBR_
0000000.h5
input file: GW1AM2_201207261145_055A_L1SGBTBR_000000
0.h5
GeophysicalName: Brightness Temperature
GranuleID: GW1AM2_201207261145_055A_L1SGBTBR_0000000
ObservationStartDateTime: 2012-07-26T11:45:43.018Z
EquatorCrossingDateTime: 2012-07-26T12:12:37.848Z
ObservationEndDateTime: 2012-07-26T12:35:09.735Z
NumberOfScans:          1979
limit of NumberOfScans =   2200
OverlapScans:            20
CoRegistrationParameterA1: 6G-1.25000,7G-1.00000,10G-
1.25000,18G-1.25000,23G-1.25000,36G-1.00000
CoRegistrationParameterA2: 6G-0.00000,7G--0.10000,10G
--0.25000,18G-0.00000,23G--0.25000,36G-0.00000
time(scan=1): 2012/07/26 11:45:43
latlon89a(pixel=1,scan=1): ( -73.3289,  136.7714)
latlon89b(pixel=1,scan=1): ( -73.4038,  137.1498)
latlonlm(pixel=1,scan=1): ( -73.3538,  136.6228)
latlon06(pixel=1,scan=1): ( -73.3592,  136.6213)
latlon07(pixel=1,scan=1): ( -73.3497,  136.6429)
latlon10(pixel=1,scan=1): ( -73.3506,  136.6001)
latlon18(pixel=1,scan=1): ( -73.3592,  136.6213)
latlon23(pixel=1,scan=1): ( -73.3506,  136.6001)
latlon36(pixel=1,scan=1): ( -73.3532,  136.6514)
tb06h(pixel=1,scan=1):     173.28
tb06v(pixel=1,scan=1):     208.22
tb07h(pixel=1,scan=1):     173.07
tb07v(pixel=1,scan=1):     207.54
tb10h(pixel=1,scan=1):     170.94
tb10v(pixel=1,scan=1):     204.95
tb18h(pixel=1,scan=1):     164.85
tb18v(pixel=1,scan=1):     199.84
tb23h(pixel=1,scan=1):     163.22
tb23v(pixel=1,scan=1):     196.53
tb36h(pixel=1,scan=1):     156.56
tb36v(pixel=1,scan=1):     186.39
tb89ah(pixel=1,scan=1):     163.76
tb89av(pixel=1,scan=1):     179.27
tb89bh(pixel=1,scan=1):     170.60
tb89bv(pixel=1,scan=1):     188.16
```

```
pdq06h(pixel=1,scan=1):    0
pdq06v(pixel=1,scan=1):    0
pdq07h(pixel=1,scan=1):    0
pdq07v(pixel=1,scan=1):    0
pdq10h(pixel=1,scan=1):    0
pdq10v(pixel=1,scan=1):    0
pdq18h(pixel=1,scan=1):    0
pdq18v(pixel=1,scan=1):    0
pdq23h(pixel=1,scan=1):    0
pdq23v(pixel=1,scan=1):    0
pdq36h(pixel=1,scan=1):    0
pdq36v(pixel=1,scan=1):    0
pdq89ah(pixel=1,scan=1):    0
pdq89av(pixel=1,scan=1):    0
pdq89bh(pixel=1,scan=1):    0
pdq89bv(pixel=1,scan=1):    0
lof06(pixel=1,scan=1): 100
lof07(pixel=1,scan=1): 100
lof10(pixel=1,scan=1): 100
lof18(pixel=1,scan=1): 100
lof23(pixel=1,scan=1): 100
lof36(pixel=1,scan=1): 100
lof89a(pixel=1,scan=1): 100
lof89b(pixel=1,scan=1): 100
ear_in(pixel=1,scan=1):      55.20
ear_az(pixel=1,scan=1):     144.76
```

６．２ L1R データ読込み

６．２．１ サンプルプログラム readL1R_amtk.f 解説

サンプルプログラム readL1R_amtk.f では、L1R データファイルから、以下のメタデータと格納データを読込んで、89G A ホーン緯度経度から低周波用緯度経度を抽出し、内容をテキスト表示します。
このサンプルプログラムでは、格納されている L1R 輝度温度の一部のみ取扱います。L1R 輝度温度データ一覧については、P.14「３．８　L1 リサンプリングデータ」を参照ください。

メタデータ	格納データ
* GeophysicalName - GranuleID - ObservationStartDateTime - EquatorCrossingDateTime - ObservationEndDateTime * NumberOfScans - OverlapScans - CoRegistrationParameterA1 - CoRegistrationParameterA2	* Scan Time * Latitude of Observation Point for 89A - Latitude of Observation Point for 89B - Longitude of Observation Point for 89A - Longitude of Observation Point for 89B * Brightness Temperature (res06,6.9GHz,H) - Brightness Temperature (res06,6.9GHz,V) - Brightness Temperature (res06,7.3GHz,H) - Brightness Temperature (res06,7.3GHz,V) - Brightness Temperature (res10,10.7GHz,H) - Brightness Temperature (res10,10.7GHz,V) - Brightness Temperature (res23,18.7GHz,H) - Brightness Temperature (res23,18.7GHz,V) - Brightness Temperature (res23,23.8GHz,H) - Brightness Temperature (res23,23.8GHz,V) - Brightness Temperature (res36,36.5GHz,H) - Brightness Temperature (res36,36.5GHz,V) - Brightness Temperature (res36,89.0GHz,H) - Brightness Temperature (res36,89.0GHz,V) - Brightness Temperature (original,89GHz-A,H) - Brightness Temperature (original,89GHz-A,V) - Brightness Temperature (original,89GHz-B,H) - Brightness Temperature (original,89GHz-B,V) * Pixel Data Quality 6 to 36 - Pixel Data Quality 89 * Land_Ocean Flag 6 to 36 - Land_Ocean Flag 89 * Earth Incidence - Earth Azimuth
89G A ホーン緯度経度から抽出するデータ	
- 低周波用緯度経度 → P.15 基礎知識編３．１０参照	

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の AMTK 関数は、最初に使用する際に使用説明を記載します。

メタデータ読込みには、AMTK_getMetaDataName()関数を使用します。
配列データ読込みには、出力されるデータ型によって以下の 4 つの関数を使い分けます。

- 出力が時刻構造体の場合　→ AMTK_getScanTime()関数
- 出力が緯度経度構造体の場合 → AMTK_getLatLon()関数
- 出力が real(4)型の場合　→ AMTK_get_SwathFloat()関数
- 出力が integer(4)型の場合　→ AMTK_get_SwathInt()関数

読込むデータ種類の指定は、アクセスラベルで行います。アクセスラベルとは、データ種類を指定するために AMTK で定義されている定数名です。

◇変数宣言　※左側の数字は行番号です

```
 1      program main
 2      implicit none
 3 C include
 4      include 'AMTK_f.h'
 :
 7 C fixed value
 8      integer(4),parameter::LMT=2200 ! limit of NumberOfScans
 9 C interface variable
10      integer(4) i,j        ! loop variable
11      integer(4) ret        ! return status
12      character(len=512) buf ! text buffer
13      character(len=512) fn  ! filename
14      integer(4) hnd        ! file handle
15 C meta data
16      character(len=512) geo  ! GeophysicalName
17      character(len=512) gid  ! GranuleID
18      character(len=512) tm1  ! ObservationStartD
19      character(len=512) tm2  ! EquatorCrossingDa
20      character(len=512) tm3  ! ObservationEndDat
21      integer(4) num         ! NumberOfScans
22      integer(4) ovr         ! OverlapScans
23      character(len=512) prm1 ! CoRegistrationPar
24      character(len=512) prm2 ! CoRegistrationPar
25 C array data
26      type(AM2_COMMON_SCANTIME) st(LMT) ! scantime
27      type(AM2_COMMON_LATLON) ll89ar(AM2_DEF_SNUM_HI,LMT) ! latlon for 89a
28      type(AM2_COMMON_LATLON) ll89br(AM2_DEF_SNUM_HI,LMT) ! latlon for 89b
 :
30      real(4) tb06h06(AM2_DEF_SNUM_LO,LMT) ! tb for 06h, resolution 06G
31      real(4) tb06v06(AM2_DEF_SNUM_LO,LMT) ! tb for 06v, resolution 06G
 :
49      integer(1) pdq06h(AM2_DEF_SNUM_LO,LMT)   ! pixel data quality for 06h
50      integer(1) pdq06v(AM2_DEF_SNUM_LO,LMT)   ! pixel data quality for 06v
 :
59      integer(1) lof06(AM2_DEF_SNUM_LO,LMT)   ! land ocean flag for 06
60      integer(1) lof10(AM2_DEF_SNUM_LO,LMT)   ! land ocean flag for 10
 :
66      real(4) ear_in(AM2_DEF_SNUM_LO,LMT) ! earth incidence
67      real(4) ear_az(AM2_DEF_SNUM_LO,LMT) ! earth azimuth
```

FORTRAN 用 AMTK ヘッダーをインクルードします。

標準プロダクトのスキャン数上限は 2200 で充分ですが、
準リアルプロダクトを使用する場合は、
2 周回程度繋がる場合があるので、
LMT=9000 としてください。(標準の 4 パス分程度)

AMTK 用インターフェイス変数を宣言します。

メタデータ用変数を宣言します。

物理量名
グラニュール ID
観測開始時刻
赤道通過時刻
観測終了時刻
スキャン数
オーバーラップスキャン数
相対レジストレーション係数 A1
相対レジストレーション係数 A2

時刻データには AM2_COMMON_SCANTIME 構造体を使います。
緯度経度データには AM2_COMMON_LATLON 構造体を使います。
配列の次元数はデータによって異なります。
スキャン数はパスによって若干上下するので、上限を決めて宣言しておきます。
ここでは上限として LMT=2200 を設定しています。
AM2_DEF_SNUM_HI は AMTK に定義されている定数で、高解像度ピクセル数(486)です。
AM2_DEF_SNUM_LO は AMTK に定義されている定数で、低解像度ピクセル数(243)です。

時刻
緯度経度
輝度温度
品質フラグ
陸海フラグ
観測入射角
観測方位角

配列データ用変数を宣言します。

◇開始処理　※左側の数字は行番号です

ファイルをオープンします。

hnd=AMTK_openH5(fn)
fn: オープンするファイル名を指定します
hnd: [戻り値]成功の場合はファイルハンドル値、失敗の場合は負の値

```
 76 C open
 77       hnd=AMTK_openH5(fn)
 78       if(hnd.lt.0)then
 79         write(*,'(a,a)')'AMTK_openH5 error: ',fn(1:len_trim(fn))
 80         write(*,'(a,i12)')'amtk status: ',hnd
 81         call exit(1)
 82       endif
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込みます。

ret=AMTK_getMetaDataName(hnd,met,out)
hnd: ファイルハンドル値を指定します
met: メタデータ名称を指定します
out: メタデータ内容が返されます
ret: [戻り値]成功の場合は読込んだメタデータの文字数、失敗の場合は負の値

```
 83 C read meta: GeophysicalName
 84       ret=AMTK_getMetaDataName(hnd,'GeophysicalName',geo)
 85       if(ret.lt.0)then
 86         write(*,'(a)')'AMTK_getMetaDataName error: GeophysicalName'
 87         write(*,'(a,i12)')'amtk status: ',ret
 88         call exit(1)
 89       endif
 90       write(*,'(a,a)')'GeophysicalName: ',geo(1:len_trim(geo))
```

メタデータからスキャン数を読込みます。
配列データ読込みではスキャン数の指定が必要になるので、ここで読込んでおきます。

```
126 C read meta: NumberOfScans
127       ret=AMTK_getMetaDataName(hnd,'NumberOfScans',buf)
128       if(ret.lt.0)then
129         write(*,'(a)')'AMTK_getMetaDataName error: NumberOfScans'
130         write(*,'(a,i12)')'amtk status: ',ret
131         call exit(1)
132       endif
133       read(buf(1:ret),*)num
134       write(*,'(a,i12)')'NumberOfScans: ',num
```

メタデータは全て文字として取得されるので、数値に変換しておきます。

◇時刻データ読込み　※左側の数字は行番号です

P.11 基礎知識編３． ６参照

時刻データを読込みます。
時刻データは AM2_COMMON_SCANTIME 構造体の 1 次元配列で、サイズはスキャン数です。
時刻データの読込みには、AMTK_getScanTime()関数を使用します。

ret=AMTK_getScanTime(hnd,bgn,end,out)
hnd: ファイルハンドル値を指定します
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
out: 出力データが返されます
ret: [戻り値]失敗の場合は負の値

```
171 C read array: scantime
172       ret=AMTK_getScanTime(hnd,1,num,st)
173       if(ret.lt.0)then
174         write(*,'(a)')'AMTK_getScanTime error.'
175         write(*,'(a,i12)')'amtk status: ',ret
176         call exit(1)
177       endif
178       write(*,'(a,i4.4,"/",i2.2,"/",i2.2," ",i2.2,":",i2.2,":",i2.2)')
179      +'time(scan=1): '
180      +,st(1)%year
181      +,st(1)%month
182      +,st(1)%day
183      +,st(1)%hour
184      +,st(1)%minute
185      +,st(1)%second
```

◇緯度経度データ読込み　※左側の数字は行番号です

P.15 基礎知識編３． １０参照

緯度経度データを読込みます。
緯度経度データは AM2_COMMON_LATLON 構造体の 2 次元配列で、サイズはピクセル数×スキャン数です。
緯度経度データの読込みには、AMTK_getLatLon()関数を使用します。

ret=AMTK_getLatLon(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
186 C read array: latlon for 89a altitude revised
187       ret=AMTK_getLatLon(hnd,ll89ar,1,num,AM2_LATLON_RS_89A)
188       if(ret.lt.0)then
189         write(*,'(a)')'AMTK_getLatLon error: AM2_LATLON_RS_89A'
190         write(*,'(a,i12)')'amtk status: ',ret
191         call exit(1)
192       endif
193       write(*,'(a,"(",f9.4,",",f9.4,")")')'latlon89ar(pixel=1,scan=1): '
194      +,ll89ar(1,1)%lat,ll89ar(1,1)%lon
```

◇輝度温度データ読込み　※左側の数字は行番号です

輝度温度データを読込みます。
輝度温度データは real(4)型の 2 次元配列で、サイズはピクセル数×スキャン数です。
出力が real(4)型なので、AMTK_get_SwathFloat()関数を使用します。

ret=AMTK_get_SwathFloat(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
213 C read array: tb for 06h, resolution 06G
214       ret=AMTK_get_SwathFloat(hnd,tb06h06,1,num,AM2_RES06_TB06H)
215       if(ret.lt.0)then
216         write(*,'(a)')'AMTK_get_SwathFloat error: AM2_RES06_TB06H'
217         write(*,'(a,i12)')'amtk status: ',ret
218         call exit(1)
219       endif
220       write(*,'(a,f9.2)')'tb06h06(pixel=1,scan=1): ',tb06h06(1,1)
```

◇L1 品質フラグデータ読込み　※左側の数字は行番号です

L1 品質フラグデータを読込みます。
L1 品質フラグデータは integer(4)型の 2 次元配列で、
L1 低解像度品質フラグデータのサイズは(ピクセル数*16)×スキャン数です。
L1 高解像度品質フラグデータのサイズは(ピクセル数* 8)×スキャン数です。
出力が integer(4)型なので、AMTK_get_SwathInt()関数を使用します。

ret=AMTK_get_SwathInt(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
357 C read array: pixel data quality for low
358       ret=AMTK_get_SwathInt(hnd,pdqlo,1,num,AM2_PIX_QUAL_LO)
359       if(ret.lt.0)then
360         write(*,'(a)')'AMTK_get_SwathInt error: AM2_PIX_QUAL_LO'
361         write(*,'(a,i12)')'amtk status: ',ret
362         call exit(1)
363       endif
```

L1 品質フラグデータは、各周波数（6GHz、7GHz)、偏波に対する RFI 情報がまとまって取得されます。
まとまったままでは扱い難いので、各周波数毎の配列に分割します。
格納値は 2 ビットのマスク値なので、各周波数毎に integer(1)型の 2 次元配列を用意します。
L1 品質フラグデータには RFI(Radio Frequency Interference; 電波干渉)情報が格納されています。
電波干渉の発生がない場合は 00、発生の可能性がある場合は 10、発生がある場合は 11 となります。

```
364       do j=1,num
365         do i=1,AM2_DEF_SNUM_LO
366           pdq06v(i,j)=pdqlo((i-1)*16+ 2,j)*10+pdqlo((i-1)*16+ 1,j)
367           pdq06h(i,j)=pdqlo((i-1)*16+ 4,j)*10+pdqlo((i-1)*16+ 3,j)
368           pdq07v(i,j)=pdqlo((i-1)*16+ 6,j)*10+pdqlo((i-1)*16+ 5,j)
369           pdq07h(i,j)=pdqlo((i-1)*16+ 8,j)*10+pdqlo((i-1)*16+ 7,j)
370         enddo
371       enddo
```

◇L1R 陸海フラグデータ読込み　※左側の数字は行番号です

陸海フラグデータを読込みます。
陸海フラグデータは integer(4)型の 2 次元配列で、
L1B 低解像度陸海データのサイズはピクセル数×(スキャン数*4)です。
L1B 高解像度陸海データのサイズはピクセル数×(スキャン数*2)です。
出力が integer(4)型なので、AMTK_get_SwathInt()関数を使用します。

P. 14 基礎知識編３． ９参照

```
395 C read array: land ocean flag for low
396       ret=AMTK_get_SwathInt(hnd,loflo,1,num,AM2_LOF_RES_LO)
397       if(ret.lt.0)then
398         write(*,'(a)')'AMTK_get_SwathInt error: AM2_LOF_RES_LO'
399         write(*,'(a,i12)')'amtk status: ',ret
400         call exit(1)
401       endif
```

陸海フラグデータは、解像度毎に全周波数データがまとまって取得されます。
まとまったままでは扱い難いので、各周波数毎の配列に分割します。
格納値は 0～100 のフラグ値なので、各周波数毎に integer(1)型の 2 次元配列を用意します。

```
402       do j=1,num
403         do i=1,AM2_DEF_SNUM_LO
404           lof06(i,j)=loflo(i,num*0+j)
405           lof10(i,j)=loflo(i,num*1+j)
406           lof23(i,j)=loflo(i,num*2+j)
407           lof36(i,j)=loflo(i,num*3+j)
408         enddo
409       enddo
```

◇観測入射角データ読込み　※左側の数字は行番号です

観測入射角データを読込みます。
観測入射角データは real(4)型の 2 次元配列で、サイズはピクセル数×スキャン数です。
出力が real(4)型なので、AMTK_get_SwathFloat()関数を使用します。

```
429 C read array: earth incidence
430       ret=AMTK_get_SwathFloat(hnd,ear_in,1,num,AM2_EARTH_INC)
431       if(ret.lt.0)then
432         write(*,'(a)')'AMTK_get_SwathFloat error: AM2_EARTH_INC'
433         write(*,'(a,i12)')'amtk status: ',ret
434         call exit(1)
435       endif
436       write(*,'(a,f9.2)')'ear_in(pixel=1,scan=1): ',ear_in(1,1)
```

◇終了処理　※左側の数字は行番号です

ファイルをクローズします。

ret=AMTK_closeH5(hnd)
hnd: クローズするファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

```
445 C close
446       ret=AMTK_closeH5(hnd)
```

６．２．２ コンパイル方法(build_readL1R_amtk_f.sh 解説)

コンパイルに使用するスクリプト build_readL1R_amtk_f.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～9 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 AMTK=/home/user1/util/AMTK_AMSR2_1.11
 8 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 9 SZIP=/home/user1/util/szip_2.1
10
```

12 行目には使用するコンパイラを指定します。
インテルコンパイラ(ifort)または PG コンパイラ(pgf90)を指定します。

```
11 # compiler
12 fc=ifort
13
14 # source filename
15 fsrc=readL1R_amtk.f
16
17 # output filename
18 out=readL1R_amtk_f
19
20 # library order
21 lib="-lAMSR2 -lhdf5 -lsz -lz -lm"
22
23 # compile
24 cmd="$fc -g $fsrc -o $out -I$AMTK/include -I$HDF5/include -I$SZIP/include
-L$AMTK/lib -L$HDF5/lib -L$SZIP/lib $lib"
25 echo $cmd
26 $cmd
27
28 # garbage
29 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL1R_amtk_f.sh
ifort -g readL1R_amtk.f -o readL1R_amtk_f
-I/home/user1/util/AMTK_AMSR2_1.11/include
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/AMTK_AMSR2_1.11/lib
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lAMSR2 -lhdf5 -lsz -lz -lm
```

６．２．３ プログラム実行結果のサンプル

サンプルプログラムでは固定配列を多数使用しているため、環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞

```
$ unlimit
```

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、4 つとも実行してください。

```
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited
```

```
$ ./readL1R_amtk_f GW1AM2_201207261145_055A_L1SGRTBR_
0000000.h5
input file: GW1AM2_201207261145_055A_L1SGRTBR_000000
0.h5
GeophysicalName: Brightness Temperature
GranuleID: GW1AM2_201207261145_055A_L1SGRTBR_0000000
ObservationStartDateTime: 2012-07-26T11:45:43.018Z
EquatorCrossingDateTime: 2012-07-26T12:12:37.848Z
ObservationEndDateTime: 2012-07-26T12:35:09.735Z
NumberOfScans:          1979
limit of NumberOfScans =   2200
OverlapScans:            20
CoRegistrationParameterA1: 6G-0.00000,7G-0.00000,10G-
0.00000,18G-0.00000,23G-0.00000,36G-0.00000
CoRegistrationParameterA2: 6G-0.00000,7G-0.00000,10G-
0.00000,18G-0.00000,23G-0.00000,36G-0.00000
time(scan=1): 2012/07/26 11:45:43
latlon89ar(pixel=1,scan=1): ( -73.3581, 136.8432)
latlon89br(pixel=1,scan=1): ( -73.4328, 137.2216)
latlonlr(pixel=1,scan=1): ( -73.3581, 136.8432)
tb06h06(pixel=1,scan=1):     173.41
tb06v06(pixel=1,scan=1):     208.09
tb07h06(pixel=1,scan=1):     173.07
tb07v06(pixel=1,scan=1):     207.11
tb10h10(pixel=1,scan=1):     170.40
tb10v10(pixel=1,scan=1):     204.58
tb18h23(pixel=1,scan=1):     165.83
tb18v23(pixel=1,scan=1):     199.41
tb23h23(pixel=1,scan=1):     163.55
tb23v23(pixel=1,scan=1):     195.90
tb36h36(pixel=1,scan=1):     153.55
tb36v36(pixel=1,scan=1):     183.95
tb89h36(pixel=1,scan=1):     163.86
tb89v36(pixel=1,scan=1):     181.05
tb89ah(pixel=1,scan=1):     163.76
tb89av(pixel=1,scan=1):     179.27
tb89bh(pixel=1,scan=1):     170.60
tb89bv(pixel=1,scan=1):     188.16
```

```
pdq06h(pixel=1,scan=1):    0
pdq06v(pixel=1,scan=1):    0
pdq07h(pixel=1,scan=1):    0
pdq07v(pixel=1,scan=1):    0
pdq10h(pixel=1,scan=1):    0
pdq10v(pixel=1,scan=1):    0
pdq18h(pixel=1,scan=1):    0
pdq18v(pixel=1,scan=1):    0
pdq23h(pixel=1,scan=1):    0
pdq23v(pixel=1,scan=1):    0
pdq36h(pixel=1,scan=1):    0
pdq36v(pixel=1,scan=1):    0
pdq89ah(pixel=1,scan=1):    0
pdq89av(pixel=1,scan=1):    0
pdq89bh(pixel=1,scan=1):    0
pdq89bv(pixel=1,scan=1):    0
lof06(pixel=1,scan=1): 100
lof10(pixel=1,scan=1): 100
lof23(pixel=1,scan=1): 100
lof36(pixel=1,scan=1): 100
lof89a(pixel=1,scan=1): 100
lof89b(pixel=1,scan=1): 100
ear_in(pixel=1,scan=1):      55.20
ear_az(pixel=1,scan=1):     144.76
```

６．３ L2 低解像度データ読込み

積算雲水量(CLW)・海氷密接度(SIC)・土壌水分量(SMC)・積雪深(SND)・海面水温(SST)・海上風速(SSW)・可降水量(TPW)が L2 低解像度です。降水量(PRC)は L2 高解像度を参照ください。

６．３．１ サンプルプログラム readL2L_amtk.f 解説

サンプルプログラム readL2L_amtk.f では、L2 低解像度データファイルから、以下のメタデータと格納データを読込んで、内容をテキスト表示します。

メタデータ	格納データ
* GeophysicalName - GranuleID - ObservationStartDateTime - EquatorCrossingDateTime - ObservationEndDateTime * NumberOfScans - OverlapScans	* Scan Time * Latitude of Observation Point * Longitude of Observation Point * Geophysical Data * Pixel Data Quality P.15 基礎知識編３．１０参照

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の AMTK 関数は、最初に使用する際に使用説明を記載します。

メタデータ読込みには、AMTK_getMetaDataName()関数を使用します。
配列データ読込みには、出力されるデータ型によって以下の 4 つの関数を使い分けます。

- 出力が時刻構造体の場合 → AMTK_getScanTime()関数
- 出力が緯度経度構造体の場合 → AMTK_getLatLon()関数
- 出力が real(4)型の場合 → AMTK_get_SwathFloat()関数
- 出力が integer(1)型の場合 → AMTK_get_SwathUChar()関数

読込むデータ種類の指定は、アクセスラベルで行います。アクセスラベルとは、データ種類を指定するために AMTK で定義されている定数名です。

◇変数宣言　※左側の数字は行番号です

```
 1      program main
 2      implicit none
 3 C include
 4      include 'AMTK_f.h'
 :


 7 C fixed value
 8      integer(4),parameter::LMT=2200 ! limit of NumberOfScans
 :
12 C interface variable
13      integer(4) i,j        ! loop variable
14      integer(4) ret        ! return status
15      character(len=512) buf ! text buffer
16      character(len=512) fn  ! filename
17      integer(4) hnd        ! file handle
18 C meta data
19      character(len=512) geo  ! GeophysicalName
20      character(len=512) gid  ! GranuleID
21      character(len=512) tm1  ! ObservationStartD
22      character(len=512) tm2  ! EquatorCrossingDa
23      character(len=512) tm3  ! ObservationEndDat
24      integer(4) num         ! NumberOfScans
25      integer(4) ovr         ! OverlapScans


26 C array data
27      type(AM2_COMMON_SCANTIME) st(LMT) ! scantime
28      type(AM2_COMMON_LATLON) ll(AM2_DEF_SNUM_LO,LMT) ! latlon

29      real(4) geo1(AM2_DEF_SNUM_LO,LMT) ! geophysical data layer 1
30      real(4) geo2(AM2_DEF_SNUM_LO,LMT) ! geophysical data layer 2

31      integer(4) pdq1(AM2_DEF_SNUM_LO,LMT) ! pixel data quality layer 1
32      integer(4) pdq2(AM2_DEF_SNUM_LO,LMT) ! pixel data quality layer 2
33      integer(1) pdqtmp(AM2_DEF_SNUM_LO,LMT*2) ! pixel data quality temporary
```

FORTRAN 用 AMTK ヘッダーをインクルードします。

標準プロダクトのスキャン数上限は 2200 で充分ですが、
準リアルプロダクトを使用する場合は、
2 周回程度繋がる場合があるので、
LMT=9000 としてください。(標準の 4 パス分程度)

AMTK 用インターフェイス変数を宣言します。

メタデータ用変数を宣言します。

物理量名
グラニュール ID
観測開始時刻
赤道通過時刻
観測終了時刻
スキャン数
オーバーラップスキャン数

時刻データには AM2_COMMON_SCANTIME 構造体を使います。
緯度経度データには AM2_COMMON_LATLON 構造体を使います。
配列の次元数はデータによって異なります。
スキャン数はパスによって若干上下するので、上限を決めて宣言しておきます。
ここでは上限として LMT=2200 を設定しています。
AM2_DEF_SNUM_HI は AMTK に定義されている定数で、高解像度ピクセル数(486)です。
AM2_DEF_SNUM_LO は AMTK に定義されている定数で、低解像度ピクセル数(243)です。

時刻
緯度経度
物理量
品質フラグ

配列データ用変数を宣言します。

◇開始処理　※左側の数字は行番号です

ファイルをオープンします。

hnd=AMTK_openH5(fn)
fn: オープンするファイル名を指定します
hnd: [戻り値]成功の場合はファイルハンドル値、失敗の場合は負の値

```
 42 C open
 43      hnd=AMTK_openH5(fn)
 44      if(hnd.lt.0)then
 45        write(*,'(a,a)')'AMTK_openH5 error: ',fn(1:len_trim(fn))
 46        write(*,'(a,i12)')'amtk status: ',hnd
 47        call exit(1)
 48      endif
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込みます。

ret=AMTK_getMetaDataName(hnd,met,out)
hnd: ファイルハンドル値を指定します
met: メタデータ名称を指定します
out: メタデータ内容が返されます
ret: [戻り値]成功の場合は読込んだメタデータの文字数、失敗の場合は負の値

```
 49 C read meta: GeophysicalName
 50      ret=AMTK_getMetaDataName(hnd,'GeophysicalName',geo)
 51      if(ret.lt.0)then
 52        write(*,'(a)')'AMTK_getMetaDataName error: GeophysicalName'
 53        write(*,'(a,i12)')'amtk status: ',ret
 54        call exit(1)
 55      endif
 56      write(*,'(a,a)')'GeophysicalName: ',geo(1:len_trim(geo))
```

メタデータからスキャン数を読込みます。
配列データ読込みではスキャン数の指定が必要になるので、ここで読込んでおきます。

```
103 C read meta: NumberOfScans
104      ret=AMTK_getMetaDataName(hnd,'NumberOfScans',buf)
105      if(ret.lt.0)then
106        write(*,'(a)')'AMTK_getMetaDataName error: NumberOfScans'
107        write(*,'(a,i12)')'amtk status: ',ret
108        call exit(1)
109      endif
110      read(buf(1:ret),*)num
111      write(*,'(a,i12)')'NumberOfScans: ',num
```

メタデータは全て文字として取得されるので、数値に変換しておきます。

◇時刻データ読込み　※左側の数字は行番号です

P.11 基礎知識編３． ６参照

時刻データを読込みます。
時刻データは AM2_COMMON_SCANTIME 構造体の 1 次元配列で、サイズはスキャン数です。
時刻データの読込みには、AMTK_getScanTime()関数を使用します。

ret=AMTK_getScanTime(hnd,bgn,end,out)
hnd: ファイルハンドル値を指定します
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
out: 出力データが返されます
ret: [戻り値]失敗の場合は負の値

```
128 C read array: scantime
129       ret=AMTK_getScanTime(hnd,1,num,st)
130       if(ret.lt.0)then
131         write(*,'(a)')'AMTK_getScanTime error.'
132         write(*,'(a,i12)')'amtk status: ',ret
133         call exit(1)
134       endif
135       write(*,'(a,i4.4,"/",i2.2,"/",i2.2," ",i2.2,":",i2.2,":",i2.2)')
136      +'time(scan=1): '
137      +,st(1)%year
138      +,st(1)%month
139      +,st(1)%day
140      +,st(1)%hour
141      +,st(1)%minute
142      +,st(1)%second
```

◇緯度経度データ読込み　※左側の数字は行番号です

P.15 基礎知識編３． １０参照

緯度経度データを読込みます。
緯度経度データは AM2_COMMON_LATLON 構造体の 2 次元配列で、サイズはピクセル数×スキャン数です。
緯度経度データの読込みには、AMTK_getLatLon()関数を使用します。

ret=AMTK_getLatLon(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
143 C read array: latlon
144       ret=AMTK_getLatLon(hnd,ll,1,num,AM2_LATLON_L2_LO)
145       if(ret.lt.0)then
146         write(*,'(a)')'AMTK_getLatLon error: AM2_LATLON_L2_LO'
147         write(*,'(a,i12)')'amtk status: ',ret
148         call exit(1)
149       endif
150       write(*,'(a,"(",f9.4,",",f9.4,")")')'latlon(pixel=1,scan=1): '
151      +,ll(1,1)%lat,ll(1,1)%lon
```

◇物理量データ読込み　※左側の数字は行番号です

物理量データを読込みます。
物理量データは real(4)型の 3 次元配列で、サイズはレイヤ数×ピクセル数×スキャン数です。
出力が real(4)型なので、AMTK_get_SwathFloat()関数を使用します。
積雪深(SND)・海面水温(SST)は物理量が 2 層あるので、この 2 つとそれ以外で処理を分けています。
積雪深の 2 層目は積雪水量[cm]です。海面水温の 2 層目は 10GHz による SST[℃]です。

ret=AMTK_get_SwathFloat(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
152 C read array: geophysical data for 1 layer
153       if((gid(30:32).ne.'SND').and.(gid(30:32).ne.'SST'))then
154         ret=AMTK_get_SwathFloat(hnd,geo1,1,num,AM2_SWATH_GEO1)
155         if(ret.lt.0)then
156           write(*,'(a)')'AMTK_get_SwathFloat error: AM2_SWATH_GEO1'
157           write(*,'(a,i12)')'amtk status: ',ret
158           call exit(1)
159         endif
160       endif
```

2 層ある場合は、アクセスラベルを変更して読み分けます。

```
161 C read array: geophysical data for 2 layer
162       if((gid(30:32).eq.'SND').or.(gid(30:32).eq.'SST'))then
163         ! layer 1
164         ret=AMTK_get_SwathFloat(hnd,geo1,1,num,AM2_SWATH_GEO1)
165         if(ret.lt.0)then
166           write(*,'(a)')'AMTK_get_SwathFloat error: AM2_SWATH_GEO1'
167           write(*,'(a,i12)')'amtk status: ',ret
168           call exit(1)
169         endif
170         ! layer 2
171         ret=AMTK_get_SwathFloat(hnd,geo2,1,num,AM2_SWATH_GEO2)
172         if(ret.lt.0)then
173           write(*,'(a)')'AMTK_get_SwathFloat error: AM2_SWATH_GEO2'
174           write(*,'(a,i12)')'amtk status: ',ret
175           call exit(1)
176         endif
177       endif
```

◇L2 品質フラグデータ読込み　※左側の数字は行番号です

L2 品質フラグデータを読込みます。
L2 品質フラグデータは integer(1)型の 3 次元配列で、
サイズはピクセル数×スキャン数×レイヤ数です。
出力が integer(1)型なので、AMTK_get_SwathUChar()関数を使用します。
積雪深(SND)、海面水温(SST)は品質フラグが 2 層あるのでこの 2 つとそれ以外で処理を分けています。

ret=AMTK_get_SwathUChar(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
178 C read array: pixel data quality for 1 layer
179       if((gid(30:32).ne.'SND').and.(gid(30:32).ne.'SST'))then
180         ! read
181         ret=AMTK_get_SwathUChar(hnd,pdqtmp,1,num,AM2_PIX_QUAL)
182         if(ret.lt.0)then
183           write(*,'(a)')'AMTK_get_SwathUChar error: AM2_PIX_QUAL'
184           write(*,'(a,i12)')'amtk status: ',ret
185           call exit(1)
186         endif
187         ! convert signed to unsigned
188         do j=1,num
189           do i=1,AM2_DEF_SNUM_LO
190             pdq1(i,j)=pdqtmp(i,j)
191             if(pdq1(i,j).ge.0)then
192               pdq1(i,j)=pdq1(i,j)
193             else
194               pdq1(i,j)=pdq1(i,j)+256
195             endif
196           enddo
197         enddo
198       endif
```

FORTRAN には符号なし変数型がないので、
L2 品質フラグ 0～255 は、
-128～127 として読込まれます。
元の値に戻すため、以下の操作を行います。
0～127→そのままの値
-128～-1→256 を足す

L2 品質フラグには、アルゴリズム開発者が設定した物理量算出に係る補足情報が格納されています。
0～15 は OK 状態、16～255 は NG 状態を現します。
NG 状態の場合、物理量には欠損値(-32768)または異常値(-32767)が格納されています。
L2 品質フラグの詳細については、「AMSR2 高次プロダクトフォーマット説明書」(*1)を参照ください。
(*1)http://suzaku.eorc.jaxa.jp/GCOM_W/data/data_w_format_j.html

2 層ある場合は、一時変数で全体を読込んだ後に、層ごとに分割しています。

```
199 C read array: pixel data quality for 2 layer
200       if((gid(30:32).eq.'SND').or.(gid(30:32).eq.'SST'))then
201         ! read
202         ret=AMTK_get_SwathUChar(hnd,pdqtmp,1,num,AM2_PIX_QUAL)
203         if(ret.lt.0)then
204           write(*,'(a)')'AMTK_get_SwathUChar error: AM2_PIX_QUAL'
205           write(*,'(a,i12)')'amtk status: ',ret
206           call exit(1)
207         endif
208         ! separate & convert signed to unsigned
209         do j=1,num
210           do i=1,AM2_DEF_SNUM_LO
211             pdq1(i,j)=pdqtmp(i,num*0+j)
212             pdq2(i,j)=pdqtmp(i,num*1+j)
213             if(pdq1(i,j).ge.0)then
214               pdq1(i,j)=pdq1(i,j)
215             else
216               pdq1(i,j)=pdq1(i,j)+256
217             endif
218             if(pdq2(i,j).ge.0)then
219               pdq2(i,j)=pdq2(i,j)
220             else
221               pdq2(i,j)=pdq2(i,j)+256
222             endif
223           enddo
224         enddo
225       endif
```

◇終了処理　※左側の数字は行番号です

ファイルをクローズします。

ret=AMTK_closeH5(hnd)
hnd: クローズするファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

```
257 C close
258       ret=AMTK_closeH5(hnd)
```

６．３．２ コンパイル方法(build_readL2L_amtk_f.sh 解説)

コンパイルに使用するスクリプト build_readL2L_amtk_f.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～9 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 AMTK=/home/user1/util/AMTK_AMSR2_1.11
 8 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 9 SZIP=/home/user1/util/szip_2.1
10
```

12 行目には使用するコンパイラを指定します。
インテルコンパイラ(ifort)または PG コンパイラ(pgf90)を指定します。

```
11 # compiler
12 fc=ifort
13
14 # source filename
15 fsrc=readL2L_amtk.f
16
17 # output filename
18 out=readL2L_amtk_f
19
20 # library order
21 lib="-lAMSR2 -lhdf5 -lsz -lz -lm"
22
23 # compile
24 cmd="$fc -g $fsrc -o $out -I$AMTK/include -I$HDF5/include -I$SZIP/include
-L$AMTK/lib -L$HDF5/lib -L$SZIP/lib $lib"
25 echo $cmd
26 $cmd
27
28 # garbage
29 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL2L_amtk_f.sh
ifort -g readL2L_amtk.f -o readL2L_amtk_f
-I/home/user1/util/AMTK_AMSR2_1.11/include
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/AMTK_AMSR2_1.11/lib
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lAMSR2 -lhdf5 -lsz -lz -lm
```

６．３．３ プログラム実行結果のサンプル

サンプルプログラムでは固定配列を多数使用しているため、環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞
```
$ unlimit
```

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、4 つとも実行してください。
```
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited
```

```
$ ./readL2L_amtk_f GW1AM2_201303011809_125D_L2SGCLWLA0000000.h5
input file: GW1AM2_201303011809_125D_L2SGCLWLA0000000.h5
GeophysicalName: Cloud Liquid Water
GranuleID: GW1AM2_201303011809_125D_L2SGCLWLA0000000
ObservationStartDateTime: 2013-03-01T18:09:10.122Z
EquatorCrossingDateTime: 2013-03-01T18:35:54.849Z
ObservationEndDateTime: 2013-03-01T18:58:26.342Z
NumberOfScans:         1972
limit of NumberOfScans =   2200
OverlapScans:             0
time(scan=1): 2013/03/01 18:09:10
latlon(pixel=1,scan=1): (  84.4574, -78.1076)
geo1(pixel=1,scan=1): -32767.000 [Kg/m2] (PDQ:112)
```

６．４ L2 高解像度データ読込み

降水量(PRC)が L2 高解像度です。
積算雲水量(CLW)・海氷密接度(SIC)・土壌水分量(SMC)・積雪深(SND)・海面水温(SST)・海上風速(SSW)・可降水量(TPW)は L2 低解像度を参照ください。

６．４．１ サンプルプログラム readL2H_amtk.f 解説

サンプルプログラム readL2H_amtk.f では、L2 高解像度データファイルから、以下のメタデータと格納データを読込んで、内容をテキスト表示します。

メタデータ	格納データ
* GeophysicalName - GranuleID - ObservationStartDateTime - EquatorCrossingDateTime - ObservationEndDateTime * NumberOfScans - OverlapScans	* Scan Time * Latitude of Observation Point for 89A - Latitude of Observation Point for 89B * Longitude of Observation Point for 89A - Longitude of Observation Point for 89B * Geophysical Data for 89A - Geophysical Data for 89B * Pixel Data Quality for 89A - Pixel Data Quality for 89B P.15 基礎知識編３．１０参照

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の AMTK 関数は、最初に使用する際に使用説明を記載します。

メタデータ読込みには、AMTK_getMetaDataName()関数を使用します。
配列データ読込みには、出力されるデータ型によって以下の４つの関数を使い分けます。

- 出力が時刻構造体の場合 → AMTK_getScanTime()関数
- 出力が緯度経度構造体の場合 → AMTK_getLatLon()関数
- 出力が real(4)型の場合 → AMTK_get_SwathFloat()関数
- 出力が integer(1)型の場合 → AMTK_get_SwathUChar()関数

読込むデータ種類の指定は、アクセスラベルで行います。アクセスラベルとは、データ種類を指定するために AMTK で定義されている定数名です。

◇変数宣言　※左側の数字は行番号です

```
 1      program main
 2      implicit none
 3 C include
 4      include 'AMTK_f.h'
 :
 7 C fixed value
 8      integer(4),parameter::LMT=2200 ! limit of NumberOfScans
 :
11 C interface variable
12      integer(4) i,j          ! loop variable
13      integer(4) ret          ! return status
14      character(len=512) buf ! text buffer
15      character(len=512) fn  ! filename
16      integer(4) hnd          ! file handle
17 C meta data
18      character(len=512) geo  ! GeophysicalName
19      character(len=512) gid  ! GranuleID
20      character(len=512) tm1  ! ObservationStartD
21      character(len=512) tm2  ! EquatorCrossingDa
22      character(len=512) tm3  ! ObservationEndDat
23      integer(4) num          ! NumberOfScans
24      integer(4) ovr          ! OverlapScans
25 C array data
26      type(AM2_COMMON_SCANTIME) st(LMT) ! scantime
27      type(AM2_COMMON_LATLON) ll89a(AM2_DEF_SNUM_HI,LMT)
28      type(AM2_COMMON_LATLON) ll89b(AM2_DEF_SNUM_HI,LMT)
29      real(4) geo1_89a(AM2_DEF_SNUM_HI,LMT)
30      real(4) geo1_89b(AM2_DEF_SNUM_HI,LMT)
31      integer(4) pdq1_89a(AM2_DEF_SNUM_HI,LMT)
32      integer(4) pdq1_89b(AM2_DEF_SNUM_HI,LMT)
33      integer(1) pdqtmp(AM2_DEF_SNUM_HI,LMT)
```

FORTRAN 用 AMTK ヘッダーをインクルードします。

標準プロダクトのスキャン数上限は 2200 で充分ですが、
準リアルプロダクトを使用する場合は、
2 周回程度繋がる場合があるので、
LMT=9000 としてください。(標準の 4 パス分程度)

AMTK 用インターフェイス変数を宣言します。

メタデータ用変数を宣言します。

物理量名
グラニュール ID
観測開始時刻
赤道通過時刻
観測終了時刻
スキャン数
オーバーラップスキャン数

時刻データには AM2_COMMON_SCANTIME 構造体を使います。
緯度経度データには AM2_COMMON_LATLON 構造体を使います。
配列の次元数はデータによって異なります。
スキャン数はパスによって若干上下するので、上限を決めて宣言しておきます。
ここでは上限として LMT=2200 を設定しています。
AM2_DEF_SNUM_HI は AMTK に定義されている定数で、高解像度ピクセル数(486)です。
AM2_DEF_SNUM_LO は AMTK に定義されている定数で、低解像度ピクセル数(243)です。

時刻
緯度経度
物理量
品質フラグ

配列データ用変数を宣言します。

◇開始処理　※左側の数字は行番号です

ファイルをオープンします。

hnd=AMTK_openH5(fn)
fn: オープンするファイル名を指定します
hnd: [戻り値]成功の場合はファイルハンドル値、失敗の場合は負の値

```
 42 C open
 43       hnd=AMTK_openH5(fn)
 44       if(hnd.lt.0)then
 45         write(*,'(a,a)')'AMTK_openH5 error: ',fn(1:len_trim(fn))
 46         write(*,'(a,i12)')'amtk status: ',hnd
 47         call exit(1)
 48       endif
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込みます。

ret=AMTK_getMetaDataName(hnd,met,out)
hnd: ファイルハンドル値を指定します
met: メタデータ名称を指定します
out: メタデータ内容が返されます
ret: [戻り値]成功の場合は読込んだメタデータの文字数、失敗の場合は負の値

```
 49 C read meta: GeophysicalName
 50       ret=AMTK_getMetaDataName(hnd,'GeophysicalName',geo)
 51       if(ret.lt.0)then
 52         write(*,'(a)')'AMTK_getMetaDataName error: GeophysicalName'
 53         write(*,'(a,i12)')'amtk status: ',ret
 54         call exit(1)
 55       endif
 56       write(*,'(a,a)')'GeophysicalName: ',geo(1:len_trim(geo))
```

メタデータからスキャン数を読込みます。
配列データ読込みではスキャン数の指定が必要になるので、ここで読込んでおきます。

```
 97 C read meta: NumberOfScans
 98       ret=AMTK_getMetaDataName(hnd,'NumberOfScans',buf)
 99       if(ret.lt.0)then
100         write(*,'(a)')'AMTK_getMetaDataName error: NumberOfScans'
101         write(*,'(a,i12)')'amtk status: ',ret
102         call exit(1)
103       endif
104       read(buf(1:ret),*)num
105       write(*,'(a,i12)')'NumberOfScans: ',num
```

メタデータは全て文字として取得されるので、数値に変換しておきます。

◇時刻データ読込み　※左側の数字は行番号です

P.11 基礎知識編３． ６参照

時刻データを読込みます。
時刻データは AM2_COMMON_SCANTIME 構造体の 1 次元配列で、サイズはスキャン数です。
時刻データの読込みには、AMTK_getScanTime()関数を使用します。

ret=AMTK_getScanTime(hnd,bgn,end,out)
hnd: ファイルハンドル値を指定します
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
out: 出力データが返されます
ret: [戻り値]失敗の場合は負の値

```
122 C read array: scantime
123       ret=AMTK_getScanTime(hnd,1,num,st)
124       if(ret.lt.0)then
125         write(*,'(a)')'AMTK_getScanTime error.'
126         write(*,'(a,i12)')'amtk status: ',ret
127         call exit(1)
128       endif
129       write(*,'(a,i4.4,"/",i2.2,"/",i2.2," ",i2.2,":",i2.2,":",i2.2)')
130      +'time(scan=1): '
131      +,st(1)%year
132      +,st(1)%month
133      +,st(1)%day
134      +,st(1)%hour
135      +,st(1)%minute
136      +,st(1)%second
```

◇緯度経度データ読込み　※左側の数字は行番号です

P.15 基礎知識編３． １０参照

緯度経度データを読込みます。
緯度経度データは AM2_COMMON_LATLON 構造体の 2 次元配列で、サイズはピクセル数×スキャン数です。
緯度経度データの読込みには、AMTK_getLatLon()関数を使用します。

ret=AMTK_getLatLon(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
137 C read array: latlon for 89a
138       ret=AMTK_getLatLon(hnd,ll89a,1,num,AM2_LATLON_L2_89A)
139       if(ret.lt.0)then
140         write(*,'(a)')'AMTK_getLatLon error: AM2_LATLON_L2_89A'
141         write(*,'(a,i12)')'amtk status: ',ret
142         call exit(1)
143       endif
144       write(*,'(a,"(",f9.4,",",f9.4,")")')'latlon89a(pixel=1,scan=1): '
145      +,ll89a(1,1)%lat,ll89a(1,1)%lon
```

◇物理量データ読込み　※左側の数字は行番号です

物理量データを読込みます。
物理量データは real(4)型の 3 次元配列で、サイズはレイヤ数×ピクセル数×スキャン数です。
出力が real(4)型なので、AMTK_get_SwathFloat()関数を使用します。

ret=AMTK_get_SwathFloat(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
155 C read array: geophysical data for 1 layer for 89a
156       ret=AMTK_get_SwathFloat(hnd,geo1_89a,1,num,AM2_SWATHA_GEO1)
157       if(ret.lt.0)then
158         write(*,'(a)')'AMTK_get_SwathFloat error: AM2_SWATHA_GEO1'
159         write(*,'(a,i12)')'amtk status: ',ret
160         call exit(1)
161       endif
```

◇L2 品質フラグデータ読込み　※左側の数字は行番号です

L2 品質フラグデータを読込みます。
L2 品質フラグデータは integer(1)型の 3 次元配列で、
サイズはピクセル数×スキャン数×レイヤ数です。
出力が integer(1)型なので、AMTK_get_SwathUChar()関数を使用します。

ret=AMTK_get_SwathUChar(hnd,out,bgn,end,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
bgn: 開始スキャンを指定します
end: 終了スキャンを指定します
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
169 C read array: pixel data quality for 1 layer for 89a
170       ! read
171       ret=AMTK_get_SwathUChar(hnd,pdqtmp,1,num,AM2_PIX_QUAL_A)
172       if(ret.lt.0)then
173         write(*,'(a)')'AMTK_get_SwathUChar error: AM2_PIX_QUAL_A'
174         write(*,'(a,i12)')'amtk status: ',ret
175         call exit(1)
176       endif
177       ! convert signed to unsigned
178       do j=1,num
179         do i=1,AM2_DEF_SNUM_HI
180           pdq1_89a(i,j)=pdqtmp(i,j)
181           if(pdq1_89a(i,j).ge.0)then
182             pdq1_89a(i,j)=pdq1_89a(i,j)
183           else
184             pdq1_89a(i,j)=pdq1_89a(i,j)+256
185           endif
186         enddo
187       enddo
```

FORTRAN には符号なし変数型がないので、
L2 品質フラグ 0～255 は、
-128～127 として読込まれます。
元の値に戻すため、以下の操作を行います。
0～127→そのままの値
-128～-1→256 を足す

L2 品質フラグには、アルゴリズム開発者が設定した物理量算出に係る補足情報が格納されています。
0～15 は OK 状態、16～255 は NG 状態を現します。
NG 状態の場合、物理量には欠損値(-32768)または異常値(-32767)が格納されています。
L2 品質フラグの詳細については、「AMSR2 高次プロダクトフォーマット説明書」(*1)を参照ください。
(*1)http://suzaku.eorc.jaxa.jp/GCOM_W/data/data_w_format_j.html

◇終了処理　※左側の数字は行番号です

ファイルをクローズします。

ret=AMTK_closeH5(hnd)
hnd: クローズするファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

```
257 C close
258       ret=AMTK_closeH5(hnd)
```

６．４．２ コンパイル方法(build_readL2H_amtk_f.sh 解説)

コンパイルに使用するスクリプト build_readL2H_amtk_f.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～9 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 AMTK=/home/user1/util/AMTK_AMSR2_1.11
 8 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 9 SZIP=/home/user1/util/szip_2.1
10
```

12 行目には使用するコンパイラを指定します。
インテルコンパイラ(ifort)または PG コンパイラ(pgf90)を指定します。

```
11 # compiler
12 fc=ifort
13
14 # source filename
15 fsrc=readL2H_amtk.f
16
17 # output filename
18 out=readL2H_amtk_f
19
20 # library order
21 lib="-lAMSR2 -lhdf5 -lsz -lz -lm"
22
23 # compile
24 cmd="$fc -g $fsrc -o $out -I$AMTK/include -I$HDF5/include -I$SZIP/include
-L$AMTK/lib -L$HDF5/lib -L$SZIP/lib $lib"
25 echo $cmd
26 $cmd
27
28 # garbage
29 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL2H_amtk_f.sh
ifort -g readL2H_amtk.f -o readL2H_amtk_f
-I/home/user1/util/AMTK_AMSR2_1.11/include
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/AMTK_AMSR2_1.11/lib
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lAMSR2 -lhdf5 -lsz -lz -lm
```

６．４．３ プログラム実行結果のサンプル

サンプルプログラムでは固定配列を多数使用しているため、環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞
```
$ unlimit
```

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、4 つとも実行してください。
```
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited
```

```
$ ./readL2H_amtk_f GW1AM2_201303011809_125D_L2SGPRCHA0000000.h5
input file: GW1AM2_201303011809_125D_L2SGPRCHA0000000.h5
GeophysicalName: Precipitation
GranuleID: GW1AM2_201303011809_125D_L2SGPRCHA0000000
ObservationStartDateTime: 2013-03-01T18:09:10.122Z
EquatorCrossingDateTime: 2013-03-01T18:35:54.849Z
ObservationEndDateTime: 2013-03-01T18:58:26.342Z
NumberOfScans:         1972
limit of NumberOfScans =   2200
OverlapScans:             0
time(scan=1): 2013/03/01 18:09:10
latlon89a(pixel=1,scan=1): (  84.4188, -77.9502)
latlon89b(pixel=1,scan=1): (  84.3305, -78.8925)
geo1_89a(pixel=1,scan=1): -32767.0 [mm/h] (PDQ: 16)
geo1_89b(pixel=1,scan=1): -32767.0 [mm/h] (PDQ: 16)
```

６．５ L3 輝度温度データ読込み

６．５．１ サンプルプログラム readL3B_amtk.f 解説

サンプルプログラム readL3B_amtk.f では、L3 輝度温度データファイルから、以下のメタデータと格納データを読込んで、内容をテキスト表示します。

メタデータ	格納データ
* GeophysicalName - GranuleID	* Brightness Temperature (H) - Brightness Temperature (V)

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の AMTK 関数は、最初に使用する際に使用説明を記載します。

メタデータ読込みには、AMTK_getMetaDataName()関数を使用します。
配列データ読込みには、AMTK_get_GridFloat()関数を使用します。

読込むデータ種類の指定は、アクセスラベルで行います。アクセスラベルとは、データ種類を指定するために AMTK で定義されている定数名です。

◇変数宣言　※左側の数字は行番号です

```
 1      program main
 2      implicit none
 3 C include
 4      include 'AMTK_f.h'
 :
10 C interface variable
11      integer(4) i,j        ! loop variable
12      integer(4) ret        ! return status
13      character(len=512) buf ! text buffer
14      character(len=512) fn  ! filename
15      integer(4) hnd        ! file handle
16      integer(4) siz(3)     ! array size
17      integer(4) x          ! grid size x
18      integer(4) y          ! grid size y
 :
24 C meta data
25      character(len=512) geo  ! GeophysicalName
26      character(len=512) gid  ! GranuleID

27 C array data
28      real(4),allocatable::tbH(:,:)
29      real(4),allocatable::tbV(:,:)
```

FORTRAN 用 AMTK ヘッダーをインクルードします。

AMTK 用インターフェイス変数を宣言します。

物理量名
グラニュール ID

メタデータ用変数
を宣言します。

輝度温度

配列データ用変数
を宣言します。

この時点では、配列データ用のメモリ領域は確保されていません。
後程、配列サイズを調べてから、メモリ領域の確保を行います。

◇開始処理　※左側の数字は行番号です

ファイルをオープンします。

hnd=AMTK_openH5(fn)
fn: オープンするファイル名を指定します
hnd: [戻り値]成功の場合はファイルハンドル値、失敗の場合は負の値

```
38 C open
39      hnd=AMTK_openH5(fn)
40      if(hnd.lt.0)then
41        write(*,'(a,a)')'AMTK_openH5 error: ',fn(1:len_trim(fn))
42        write(*,'(a,i12)')'amtk status: ',hnd
43        call exit(1)
44      endif
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込みます。

ret=AMTK_getMetaDataName(hnd,met,out)
hnd: ファイルハンドル値を指定します
met: メタデータ名称を指定します
out: メタデータ内容が返されます
ret: [戻り値]成功の場合は読込んだメタデータの文字数、失敗の場合は負の値

```
45 C read meta: GeophysicalName
46       ret=AMTK_getMetaDataName(hnd,'GeophysicalName',geo)
47       if(ret.lt.0)then
48         write(*,'(a)')'AMTK_getMetaDataName error: GeophysicalName'
49         write(*,'(a,i12)')'amtk status: ',ret
50         call exit(1)
51       endif
52     write(*,'(a,a)')'GeophysicalName: ',geo(1:len_trim(geo))
```

◇配列サイズ取得とメモリ領域確保　※左側の数字は行番号です

配列サイズを取得します。
配列サイズの取得には、AMTK_getDimSize()関数を使用します。

ret=AMTK_getDimSize(hnd,label,siz)
hnd: ファイルハンドル値を指定します
label: アクセスラベルを指定します
siz: [戻り値]配列サイズ
ret: [戻り値]失敗の場合は負の値

```
72 C get grid size
73       ret=AMTK_getDimSize(hnd,AM2_GRID_TBH,siz);
74       if(ret.lt.0)then
75         write(*,'(a)')'AMTK_getDimSize error: AM2_GRID_TBH'
76         write(*,'(a,i12)')'amtk status: ',ret
77         call exit(1)
78       endif
79       x=siz(2);
80       y=siz(1);
81       write(*,'(a,i12)')'grid size x: ',x
82       write(*,'(a,i12)')'grid size y: ',y
```

メモリ領域を確保します。

```
83 C memory allocate
84       allocate(tbH(x,y),stat=ret)
85       if(ret.ne.0)then
86         write(*,'(a)')'memory allocate error: tbH'
87         call exit(1)
88       endif
```

◇輝度温度読込み　※左側の数字は行番号です

輝度温度データを読込みます。

ret=AMTK_get_GridFloat(hnd,out,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
 94 C read horizontal
 95       ret=AMTK_get_GridFloat(hnd,tbH,AM2_GRID_TBH)
 96       if(ret.lt.0)then
 97         write(*,'(a)')'AMTK_get_GridFloat error: AM2_GRID_TBH'
 98         write(*,'(a,i12)')'amtk status: ',ret
 99         call exit(1)
100       endif
```

◇終了処理　※左側の数字は行番号です

メモリを開放します。

```
200 C memory free
201       deallocate(tbH)
202       deallocate(tbV)
```

ファイルをクローズします。

ret=AMTK_closeH5(hnd)
hnd: クローズするファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

```
203 C close
204       ret=AMTK_closeH5(hnd)
```

６．５．２ コンパイル方法(build_readL3B_amtk_f.sh 解説)

コンパイルに使用するスクリプト build_readL3B_amtk_f.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～9 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 AMTK=/home/user1/util/AMTK_AMSR2_1.11
 8 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 9 SZIP=/home/user1/util/szip_2.1
10
```

12 行目には使用するコンパイラを指定します。
インテルコンパイラ(ifort)または PG コンパイラ(pgf90)を指定します。

```
11 # compiler
12 fc=ifort
13
14 # source filename
15 fsrc=readL3B_amtk.f
16
17 # output filename
18 out=readL3B_amtk_f
19
20 # library order
21 lib="-lAMSR2 -lhdf5 -lsz -lz -lm"
22
23 # compile
24 cmd="$fc -g $fsrc -o $out -I$AMTK/include -I$HDF5/include -I$SZIP/include
-L$AMTK/lib -L$HDF5/lib -L$SZIP/lib $lib"
25 echo $cmd
26 $cmd
27
28 # garbage
29 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL3B_amtk_f.sh
ifort -g readL3B_amtk.f -o readL3B_amtk_f
-I/home/user1/util/AMTK_AMSR2_1.11/include
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/AMTK_AMSR2_1.11/lib
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lAMSR2 -lhdf5 -lsz -lz -lm
```

６．５．３ プログラム実行結果のサンプル

環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞
$ unlimit

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、4 つとも実行してください。
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited

```
$ ./readL3B_amtk_f GW1AM2_20130200_01M_EQMA_L3SGT06LA1110110.h5
input file: GW1AM2_20130200_01M_EQMA_L3SGT06LA1110110.h5
GeophysicalName: Brightness Temperature (6GHz)
GranuleID: GW1AM2_20130200_01M_EQMA_L3SGT06LA1110110
grid size x:         1440
grid size y:          720

ASCII ART OF HORIZONTAL BRIGHTNESS TEMPERATURE (X/ 20GRID Y/ 40GRID)
+------------------------------------------------------------------------+
|                                                                        |
|4112333444434444444444444444444444444444444444444444444444433443222222444444|
|1111343111134444444444444444444444444444444444444444444444444434233343111|
|1245234455555555444444444444444434312343444144555444444444442112111111111|
|1545444444444444444444444445441111111111111111111355444444444222111111111112|
|4122425544144445444444453411311111111111111111111455544445511111111111115|
|4444444442444555445555441111111111111111111111155555351111111111111124|
|4444444424441115551555311111111111111111121111111155151141111111111444|
|5555555555111112111131112111111111111111111111111111141554111111111155|
|1155555551111111111515531111111111111111111111111111114555554111111111|
|1115555531111111111111114211113121111111111111111111111555555551111111|
|1115555215111111111111114555551111111111111111111111111145555521111111|
|1111555111111111111111135555555111111111111111111111111154552111111111|
|1111111111111111111111111111111113111111111111111111111155111111111111|
|1111111111111111111111111111111111111111111111111111111351111111111111|
|1111111111111111111111111111111111111111111111111111111111111111111111|
|3433234443333424343343444444443343321111111111111111111344444441111133|
|3333333333333333333333333333333343333333222223333333443344333333333333333|
+------------------------------------------------------------------------+
```

```
ASCII ART OF VERTICAL BRIGHTNESS TEMPERATURE (X/ 20GRID Y/ 40GRID)
+------------------------------------------------------------------------+
|                                                                        |
|43334444444445545544444444444444444444445444444444444444444433333244545|
|33334543333455455444454444444444444444444545444444444444445544344344333|
|33553445555555555554555445455554443345455434555554444554444333333333333|
|35555555555555555545555555555433333333333333333345555555555433333333333333|
|43334355553555555555555545334333333333333333333355555555553333333333333335|
|55555555554555555555555553333333333333333333333355555453333333333333335|
|55555555355533355535554333333333333333333333333333553533433333333333555|
|65556665555333333333433343333333333333333333333333333535555333333333355|
|33555555633333333335355433333333333333333333333333333355555554333333333|
|33355555433333333333333433333433333333333333333333333355555556333333333|
|33355553353333333333333566665333333333333333333333333355555543333333333|
|33336553333333333333346666665333333333333333333333333555554333333333333|
|33333333333333333333333333333333343333333333333333333355333333333333333|
|33333333333333333333333333333333333333333333333333333345333333333333333|
|33333333333333333333333333333333333333333333333333333333333333333333333|
|44433444444444344444444444444444444433333333333333333333444555543333344|
|44444443344444444443333443444444444444444433444444444444444444444444444|
+------------------------------------------------------------------------+

[#]:missing
[ ]:out of observation
[1]: 50-100K
[2]:100-150K
[3]:150-200K
[4]:200-250K
[5]:250-300K
[6]:300-350K
[*]:other
```

６．６ L3 物理量データ読込み

６．６．１ サンプルプログラム readL3G_amtk.f 解説

サンプルプログラム readL3G_amtk.f では、L3 物理量データファイルから、以下のメタデータと格納データを読込んで、内容をテキスト表示します。

メタデータ	格納データ
* GeophysicalName - GranuleID	* Geophysical Data

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の AMTK 関数は、最初に使用する際に使用説明を記載します。

メタデータ読込みには、AMTK_getMetaDataName()関数を使用します。
配列データ読込みには、AMTK_get_GridFloat()関数を使用します。

読込むデータ種類の指定は、アクセスラベルで行います。アクセスラベルとは、データ種類を指定するために AMTK で定義されている定数名です。

◇変数宣言　※左側の数字は行番号です

```
 1      program main
 2      implicit none
 3 C include
 4      include 'AMTK_f.h'
 :
10 C interface variable
11      integer(4) i,j        ! loop variable
12      integer(4) ret        ! return status
13      character(len=512) buf ! text buffer
14      character(len=512) fn  ! filename
15      integer(4) hnd        ! file handle
16      integer(4) siz(3)     ! array size
17      integer(4) x          ! grid size x
18      integer(4) y          ! grid size y
 :
26 C meta data
27      character(len=512) geo  ! GeophysicalName
28      character(len=512) gid  ! GranuleID

29 C array data
30      real(4),allocatable::geo1(:,:)
31      real(4),allocatable::geo2(:,:)
```

FORTRAN 用 AMTK ヘッダーをインクルードします。

AMTK 用インターフェイス変数を宣言します。

物理量名
グラニュール ID

メタデータ用変数
を宣言します。

物理量

配列データ用変数
を宣言します。

この時点では、配列データ用のメモリ領域は確保されていません。
後程、配列サイズを調べてから、メモリ領域の確保を行います。

◇開始処理　※左側の数字は行番号です

ファイルをオープンします。

hnd=AMTK_openH5(fn)
fn: オープンするファイル名を指定します
hnd: [戻り値]成功の場合はファイルハンドル値、失敗の場合は負の値

```
40 C open
41      hnd=AMTK_openH5(fn)
42      if(hnd.lt.0)then
43        write(*,'(a,a)')'AMTK_openH5 error: ',fn(1:len_trim(fn))
44        write(*,'(a,i12)')'amtk status: ',hnd
45        call exit(1)
46      endif
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込みます。

ret=AMTK_getMetaDataName(hnd,met,out)
hnd: ファイルハンドル値を指定します
met: メタデータ名称を指定します
out: メタデータ内容が返されます
ret: [戻り値]成功の場合は読込んだメタデータの文字数、失敗の場合は負の値

```
47 C read meta: GeophysicalName
48       ret=AMTK_getMetaDataName(hnd,'GeophysicalName',geo)
49       if(ret.lt.0)then
50         write(*,'(a)')'AMTK_getMetaDataName error: GeophysicalName'
51         write(*,'(a,i12)')'amtk status: ',ret
52         call exit(1)
53       endif
54     write(*,'(a,a)')'GeophysicalName: ',geo(1:len_trim(geo))
```

◇配列サイズ取得とメモリ領域確保　※左側の数字は行番号です

配列サイズを取得します。
配列サイズの取得には、AMTK_getDimSize()関数を使用します。

ret=AMTK_getDimSize(hnd,label,siz)
hnd: ファイルハンドル値を指定します
label: アクセスラベルを指定します
siz: [戻り値]配列サイズ
ret: [戻り値]失敗の場合は負の値

```
75 C get grid size
76       ret=AMTK_getDimSize(hnd,AM2_GRID_GEO1,siz);
77       if(ret.lt.0)then
78         write(*,'(a)')'AMTK_getDimSize error: AM2_GRID_GEO1'
79         write(*,'(a,i12)')'amtk status: ',ret
80         call exit(1)
81       endif
82       x=siz(2);
83       y=siz(1);
84       write(*,'(a,i12)')'grid size x: ',x
85       write(*,'(a,i12)')'grid size y: ',y
```

メモリ領域を確保します。

```
86 C memory allocate layer 1
87       allocate(geo1(x,y),stat=ret)
88       if(ret.ne.0)then
89         write(*,'(a)')'memory allocate error: geo1'
90         call exit(1)
91       endif
```

◇物理量読込み　※左側の数字は行番号です

物理量データを読込みます。
積雪深(SND)・海面水温(SST)は物理量が 2 層あるので、この 2 つとそれ以外で処理を分けています。
積雪深の 2 層目は積雪水量[cm]です。海面水温の 2 層目は 10GHz による SST[℃]です。

ret=AMTK_get_GridFloat(hnd,out,label)
hnd: ファイルハンドル値を指定します
out: 出力データが返されます
label: アクセスラベルを指定します。アクセスラベルはデータ種類によって異なります
ret: [戻り値]失敗の場合は負の値

```
100 C read layer 1
101       ret=AMTK_get_GridFloat(hnd,geo1,AM2_GRID_GEO1)
102       if(ret.lt.0)then
103         write(*,'(a)')'AMTK_get_GridFloat error: AM2_GRID_GEO1'
104         write(*,'(a,i12)')'amtk status: ',ret
105         call exit(1)
106       endif
```

2 層ある場合は、アクセスラベルを変更して読み分けます。

```
107 C read layer 2
108       if(gid(30:32).eq.'SND')then
109         ret=AMTK_get_GridFloat(hnd,geo2,AM2_GRID_GEO2)
110         if(ret.lt.0)then
111           write(*,'(a)')'AMTK_get_GridFloat error: AM2_GRID_GEO2'
112           write(*,'(a,i12)')'amtk status: ',ret
113           call exit(1)
114         endif
115       endif
```

◇終了処理　※左側の数字は行番号です

メモリを開放します。

```
301 C memory free
302       deallocate(geo1)
303       if(gid(30:32).eq.'SND')then
304         deallocate(geo2)
305       endif
```

ファイルをクローズします。

ret=AMTK_closeH5(hnd)
hnd: クローズするファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

```
306 C close
307       ret=AMTK_closeH5(hnd)
```

６．６．２ コンパイル方法(build_readL3G_amtk_f.sh 解説)

コンパイルに使用するスクリプト build_readL3G_amtk_f.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～9 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 AMTK=/home/user1/util/AMTK_AMSR2_1.11
 8 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 9 SZIP=/home/user1/util/szip_2.1
10
```

12 行目には使用するコンパイラを指定します。
インテルコンパイラ(ifort)または PG コンパイラ(pgf90)を指定します。

```
11 # compiler
12 fc=ifort
13
14 # source filename
15 fsrc=readL3G_amtk.f
16
17 # output filename
18 out=readL3G_amtk_f
19
20 # library order
21 lib="-lAMSR2 -lhdf5 -lsz -lz -lm"
22
23 # compile
24 cmd="$fc -g $fsrc -o $out -I$AMTK/include -I$HDF5/include -I$SZIP/include
-L$AMTK/lib -L$HDF5/lib -L$SZIP/lib $lib"
25 echo $cmd
26 $cmd
27
28 # garbage
29 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL3G_amtk_f.sh
ifort -g readL3G_amtk.f -o readL3G_amtk_f
-I/home/user1/util/AMTK_AMSR2_1.11/include
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/AMTK_AMSR2_1.11/lib
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lAMSR2 -lhdf5 -lsz -lz -lm
```

６．６．３ プログラム実行結果のサンプル

環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞
```
$ unlimit
```

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、4 つとも実行してください。
```
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited
```

```
$ ./readL3G_amtk_f GW1AM2_20130200_01M_EQMA_L3SGCLWLA0000000.h5
input file: GW1AM2_20130200_01M_EQMA_L3SGCLWLA0000000.h5
GeophysicalName: Cloud Liquid Water
GranuleID: GW1AM2_20130200_01M_EQMA_L3SGCLWLA0000000
grid size x:         1440
grid size y:          720

ASCII ART OF GEOPHYSICAL DATA LAYER #1 (X/ 20GRID Y/ 40GRID)
+------------------------------------------------------------------------+
|                                                                        |
|#21#####################################################################|
|2212###1113#############################################################222|
|1#################################11######3################011#42223322|
|2##############################00101112413333233##############343413131#|
|#20#######2###############00#122345234421321112O#########3355421131121#|
|#########################4#32222232321001101111O#######14*41111201111##|
|########0###000###0####1111111121012011O*131121100##1#11#01211110100###|
|###########1011111111#211#11111210011121333223224321000#1###1111000000##|
|11#######00010122125#3###1#122323211000100201100010111122#######222321111|
|000######122131112311112#01112#113122123231101000000000110########1001110|
|100####32#12111141111000######223232242121244133111001101O######11111100|
|1000###121123221125110O########131112232110121313122011010#####231111001|
|11131212231110111111211111110012102#123122122331121311102l##011124223242|
|*33222511232222222223223331421022422322211024222222122113##0111421221132|
|10112211222111122211111111112223123223233124332121232322122331223222211l|
|#####################################3111221222211121122l########21221##|
|#####################################################################|
+------------------------------------------------------------------------+

[#]:missing
[ ]:out of observation
[0]:  0.000 -  0.030 Kg/m2
[1]:  0.030 -  0.060 Kg/m2
[2]:  0.060 -  0.090 Kg/m2
[3]:  0.090 -  0.120 Kg/m2
[4]:  0.120 -  0.150 Kg/m2
[5]:  0.150 -  0.180 Kg/m2
[*]:other
```

７．HDF5 編(C 言語)

赤文字の解説はサンプルプログラムについて説明しています。

青文字の解説は関数リファレンスまたは衛星基礎知識について説明しています。

７．１ L1B データ読込み

７．１．１ サンプルプログラム readL1B_hdf5.c 解説

サンプルプログラム readL1B_hdf5.c では、L1B データファイルから、以下のメタデータと格納データを読込んで、89G A ホーン緯度経度から低周波緯度経度を算出し、内容をテキスト表示します。
TAI93 形式時刻の変換と低周波緯度経度の算出には、サブルーチンを用意しています。

メタデータ	格納データ
* GeophysicalName - GranuleID - ObservationStartDateTime - EquatorCrossingDateTime - ObservationEndDateTime * NumberOfScans - OverlapScans - CoRegistrationParameterA1 - CoRegistrationParameterA2	* Scan Time * Latitude of Observation Point for 89A - Latitude of Observation Point for 89B - Longitude of Observation Point for 89A - Longitude of Observation Point for 89B * Brightness Temperature (6.9GHz,H) - Brightness Temperature (6.9GHz,V) - Brightness Temperature (7.3GHz,H) - Brightness Temperature (7.3GHz,V) - Brightness Temperature (10.7GHz,H) - Brightness Temperature (10.7GHz,V) - Brightness Temperature (18.7GHz,H) - Brightness Temperature (18.7GHz,V) - Brightness Temperature (23.8GHz,H) - Brightness Temperature (23.8GHz,V) - Brightness Temperature (36.5GHz,H) - Brightness Temperature (36.5GHz,V) - Brightness Temperature (89.0GHz-A,H) - Brightness Temperature (89.0GHz-A,V) - Brightness Temperature (89.0GHz-B,H) - Brightness Temperature (89.0GHz-B,V) * Pixel Data Quality 6 to 36 - Pixel Data Quality 89 * Land_Ocean Flag 6 to 36 - Land_Ocean Flag 89 * Earth Incidence - Earth Azimuth
89G A ホーン緯度経度から算出するデータ	
* 緯度経度(低周波平均) - 緯度経度(6G) - 緯度経度(7G) - 緯度経度(10G) - 緯度経度(18G) - 緯度経度(23G) - 緯度経度(36G) → P.15 基礎知識編３．１０参照	

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の HDF5 関数は、最初に使用する際に使用説明を記載します。

HDF5 ライブラリでは、ファイルをオープンした後に、各データについてもオープンとクローズを繰り返します。各データの読込みは以下のような流れになります。

データをオープンする →必要な場合はスケールを取得する
→ データを読込む → データをクローズする

◇変数宣言　※左側の数字は行番号です

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include "hdf5.h"
 4 #include "amsr2time.h"
 5 #include "amsr2latlon.h"
 6
 7 // fixed value
 8 #define LMT 2200 // limit of NumberOfScans
 9 #define AM2_DEF_SNUM_HI 486 // high resolution pixel width
10 #define AM2_DEF_SNUM_LO 243 // low  resolution pixel width
11
12 int main(int argc, char *argv[]){
13   // interface variable
14   int i,j;          // loop variable
15   herr_t ret;       // return status
16   char *buf = NULL; // text buffer
17   char *fn  = NULL; // filename
18   hid_t fhnd;       // file handle
19   hid_t ahnd;       // attribute handle
20   hid_t atyp;       // attribute type
21   hid_t dhnd;       // dataset handle
22   hid_t shnd;       // dataspace handle
23
24   // meta data
25   char *geo = NULL; // GeophysicalName
26   char *gid = NULL; // GranuleID
27   char *tm1 = NULL; // ObservationStartDateTime
28   char *tm2 = NULL; // EquatorCrossingDateTime
29   char *tm3 = NULL; // ObservationEndDateTime
30   int num;          // NumberOfScans
31   int ovr;          // OverlapScans
32   double prm1[7];   // CoRegistrationParameterA1
33   double prm2[7];   // CoRegistrationParameterA2

40   // array data
41   AM2_COMMON_SCANTIME st[LMT]; // scantime
42   float lat89a[LMT][AM2_DEF_SNUM_HI]; // lat for 89a
43   float lat89b[LMT][AM2_DEF_SNUM_HI]; // lat for 89b

51   float lon89a[LMT][AM2_DEF_SNUM_HI]; // lon for 89a
52   float lon89b[LMT][AM2_DEF_SNUM_HI]; // lon for 89b

60   float tb06h [LMT][AM2_DEF_SNUM_LO]; // tb for 06h
61   float tb06v [LMT][AM2_DEF_SNUM_LO]; // tb for 06v

76   unsigned char pdq06h [LMT][AM2_DEF_SNUM_LO]; // pixel data quality for 06h
77   unsigned char pdq06v [LMT][AM2_DEF_SNUM_LO]; // pixel data quality for 06v

85   unsigned char lof06 [LMT  ][AM2_DEF_SNUM_LO]; // land ocean flag for 06
86   unsigned char lof07 [LMT  ][AM2_DEF_SNUM_LO]; // land ocean flag for 07

94   float ear_in[LMT][AM2_DEF_SNUM_LO]; // earth incidence
95   float ear_az[LMT][AM2_DEF_SNUM_LO]; // earth azimuth
```

HDF5 用ヘッダーファイルをインクルードします。

時刻変換と緯度経度計算用サブルーチンヘッダーをインクルードします。

標準プロダクトのスキャン数上限は2200 で充分ですが、
準リアルプロダクトを使用する場合は、
2 周回程度繋がる場合があるので、
LMT=9000 としてください。
(標準の 4 パス分程度)

HDF5 用インターフェイス変数を宣言します。

物理量名
グラニュール ID
観測開始時刻
赤道通過時刻
観測終了時刻
スキャン数
オーバーラップスキャン数
相対レジストレーション係数 A1
相対レジストレーション係数 A2

メタデータ用変数を宣言します。

時刻データには AM2_COMMON_SCANTIME 構造体を使います。
配列の次元数はデータによって異なります。
スキャン数はパスによって若干上下するので、上限を決めて宣言しておきます。
ここでは上限として LMT=2200 を設定しています。
AM2_DEF_SNUM_HI は高解像度ピクセル数(486)です。
AM2_DEF_SNUM_LO は低解像度ピクセル数(243)です。

時刻
緯度
経度
輝度温度
品質フラグ
陸海フラグ
観測入射角
観測方位角

配列データ用変数を宣言します。

◇開始処理　※左側の数字は行番号です

HDF5 の初期化処理を行います。

◇HDF5 の初期化
ret = H5open();
ret: [戻り値]失敗の場合は負の値

```
107   // hdf5 initialize
108   ret = H5open();
109   if(ret < 0){
110     printf("h5open error: %d¥n" ,ret);
111     exit(1);
112   }
113
```

ファイルをオープンします。

◇HDF5 ファイルオープン
fhnd = H5Fopen(fn, label1, label2);
fn: オープンするファイル名を指定します
label1: アクセスモードを指定します H5F_ACC_RDONLY で読込専用になります
label2: H5P_DEFAULT を指定します
fhnd: [戻り値]開いたファイルのハンドル値、失敗の場合は負の値

```
114   // open
115   fhnd = H5Fopen(fn, H5F_ACC_RDONLY, H5P_DEFAULT);
116   if(fhnd < 0){
117     printf("H5Fopen error: %s¥n", fn);
118     exit(1);
119   }
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込むには、ファイルに付属しているアトリビュートを H5Aopen()でオープンします。
アトリビュートの型を調べてからデータを読込みます。
不要になったハンドルはクローズします。

◇アトリビュートのオープン
ahnd = H5Aopen(fhnd, nam, label);
fhnd: ファイルハンドル値を指定します
nam: アトリビュート名を指定します
label: H5P_DEFAULT を指定します
ahnd: [戻り値]アトリビュートのハンドル値
失敗の場合は負の値

◇アトリビュートタイプの取得
atyp = H5Aget_type(ahnd);
ahnd: アトリビュートハンドル値を指定します
atyp: [戻り値]アトリビュートタイプ値
失敗の場合は負の値

◇アトリビュート読込み
ret = H5Aread(ahnd, otyp, buf);
ahnd: アトリビュートハンドル値を指定します
otyp: 出力変数の型を指定します(読込んだデータは、出力変数の型へ自動変換されます)
buf: 出力変数を指定します
ret: [戻り値]失敗の場合は負の値

```
121   // read meta: GeophysicalName
122   ahnd = H5Aopen(fhnd, "GeophysicalName", H5P_DEFAULT);
123   atyp = H5Aget_type(ahnd);
124   ret = H5Aread(ahnd, atyp, &geo);
125   if(ret < 0){
126     printf("H5Aread error: GeophysicalName¥n");
127     exit(1);
128   }
129   ret = H5Aclose(ahnd);
130   printf("GeophysicalName: %s¥n", geo);
```

◇アトリビュートのクローズ
ret = H5Aclose(ahnd);
ahnd:アトリビュートハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

メタデータからスキャン数を読込みます。
配列データ読込みではスキャン数の指定が必要になります。

```
176   // read meta: NumberOfScans
177   ahnd = H5Aopen(fhnd, "NumberOfScans", H5P_DEFAULT);
178   atyp = H5Aget_type(ahnd);
179   ret = H5Aread(ahnd, atyp, &buf);
180   if(ret < 0){
181     printf("H5Aread error: NumberOfScans¥n");
182     exit(1);
183   }
184   ret = H5Aclose(ahnd);
185   num = atoi(buf);
186   printf("NumberOfScans: %d¥n", num);
```

メタデータは全て文字として取得されるので、
数値に変換しておきます。

メタデータから相対レジストレーション係数を読込みます。
相対レジストレーション係数は、低周波緯度経度を算出する際に必要になります。

```
209   // read meta: CoRegistrationParameterA1
210   ahnd = H5Aopen(fhnd, "CoRegistrationParameterA1", H5P_DEFAULT);
211   atyp = H5Aget_type(ahnd);
212   ret = H5Aread(ahnd, atyp, &buf);
213   if(ret < 0){
214     printf("H5Aread error: CoRegistrationParameterA1¥n");
215     exit(1);
216   }
217   ret = H5Aclose(ahnd);
```

P.15 基礎知識編３．１０参照

◇時刻データ読込み　※左側の数字は行番号です

データを読込むためには、データセットを H5Dopen()でオープンします。
L1 データには前後にオーバーラップが含まれるので重複部分を取り除きます。
時刻データの格納形式は TAI93 形式なので年/月/日/時/分/秒に変換します。

P.11 基礎知識編３． ５参照

◇データセットのオープン
dhnd = H5Dopen(fhnd, nam, label);
fhnd: ファイルハンドル値を指定します
nam: データセット名を指定します
label: H5P_DEFAULT を指定します
dhnd: [戻り値]データセットのハンドル値
失敗の場合は負の値

◇データ読込み
ret = H5Dread(dhnd, otyp, label1, label2, label3, buf);
dhnd: データセットハンドル値を指定します
otyp: 出力変数の型を指定します(読込んだデータは、出力変数の型へ自動変換されます)
label1,label2: 部分配列を読込む場合に使用します
全て読込む場合は、どちらも H5S_ALL を指定します
label3: H5P_DEFAULT を指定します
buf: 出力変数を指定します
ret: [戻り値]失敗の場合は負の値

◇データセットのクローズ
ret = H5Dclose(dhnd);
dhnd: データセットハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

```
265   // read array: scantime
266   dhnd = H5Dopen(fhnd, "Scan Time", H5P_DEFAULT);
267   ret = H5Dread(dhnd, H5T_NATIVE_DOUBLE, H5S_ALL, H5S_ALL, H5P_DEFAULT, r8d1);
268   if(ret < 0){
        printf("H5Dread error: Scan Time¥n");
        exit(1);
271   }
272   ret = H5Dclose(dhnd);
273   // cutoff overlap
274   for(j = 0; j < num; ++j){
275     r8d1[j] = r8d1[j + ovr];
      }
      for(j = num; j < LMT; ++j){
278     r8d1[j] = 0;
279   }
280   // convert
      amsr2time_(&num, r8d1, st);
      // sample display
283   printf("time[scan=0]: %04d/%02d/%02d %02d:%02d:%02d¥n"
284    , st[0].year
285    , st[0].month
286    , st[0].day
287    , st[0].hour
288    , st[0].minute
289    , st[0].second
290    );
```

読込み

重複除去

時刻変換

AMSR2 プロダクトの時刻格納形式は、TAI93 形式と呼ばれる、うるう秒を含めた 1993/01/01 からの通算秒です。
このままでは日時情報として扱い難いので、
このサンプルプログラムでは年/月/日/時/分/秒に変換するサブルーチンを用意しています。この時刻変換は AMTK を使用する場合は自動的に行われます。
→ P.11 基礎知識編３． ６参照

◆時刻形式の変換
amsr2time_(num,in,out)
num: スキャン数を指定します
in: TAI93 形式の時刻データを指定します
out: [戻り値]AM2_COMMON_SCANTIME 構造体で年/月/日/時/分/秒データが返されます

◇緯度経度データ読込み　※左側の数字は行番号です

89G 緯度経度は格納データを読込みます。
読込んだ後は、オーバーラップを取り除きます。

低周波緯度経度は、89G A ホーン緯度経度と相対レジストレーション係数から算出します。

緯度読込み

経度読込み

重複除去

低周波計算

```
292   // read array: latlon for 89a
293   // read lat
294   dhnd = H5Dopen(fhnd ,"Latitude of Observation Point for 89A", H5P_DEFAULT);
295   ret = H5Dread(dhnd, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL, H5P_DEFAULT, lat89a);
296   if(ret < 0){
        printf("H5Dread error: Latitude of Observation Point for 89A¥n");
        exit(1);
299   }
300   ret = H5Dclose(dhnd);
301   // read lon
302   dhnd = H5Dopen(fhnd ,"Longitude of Observation Point for 89A", H5P_DEFAULT);
303   ret = H5Dread(dhnd, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL, H5P_DEFAULT, lon89a);
304   if(ret < 0){
        printf("H5Dread error: Longitude of Observation Point for 89A¥n");
        exit(1);
307   }
308   ret = H5Dclose(dhnd);
309   // cutoff overlap
310   for(j = 0; j < num; ++j){
311     for(i = 0; i < AM2_DEF_SNUM_HI; ++i)
312       lat89a[j][i] = lat89a[j+ovr][i];
313       lon89a[j][i] = lon89a[j+ovr][i];
314     }
315   }
316   for(j = num; j < LMT; ++j){
        for(i = 0; i < AM2_DEF_SNUM_HI; ++i)
          lat89a[j][i] = 0;
          lon89a[j][i] = 0;
320     }
321   }
  :

358   // read array: latlon for low mean
      amsr2latlon_(&num, &prm1[6], &prm2[6], lat89a, lon89a, latlm, lonlm);
      printf("latlonlm[scan=0][pixel=0]: (%9.4f,%9.4f)¥n", latlm[0][0],
lonl    0][0]);
```

AMSR2 センサには 6G/7G/10G/18G/23G/36G/89G の 7 つの観測周波数がありますが、それらの観測点緯度経度は正確には異なります。
AMSR2 プロダクトに格納されているのは 89G 緯度経度だけです。低周波の正確な緯度経度は 89G A ホーン緯度経度と相対レジストレーション係数から算出できます。
このサンプルプログラムでは低周波緯度経度を算出するためのサブルーチンを用意しています。AMTK を使用する場合は、低周波緯度経度は自動的に算出されます。
→ P.15 基礎知識編３．１０参照

◆低周波緯度経度算出
amsr2latlon_(num, prm1, prm2, lat89a, lon89a, latlow, lonlow)
num: スキャン数を指定します
prm1: 相対レジストレーション係数 A1 を指定します(6G/7G/10G/18G/23G/36G/平均)
prm2: 相対レジストレーション係数 A2 を指定します(6G/7G/10G/18G/23G/36G/平均)
lat89a: 89G A ホーン緯度データを指定します
lon89a: 89G A ホーン経度データを指定します
latlow: [戻り値]指定した低周波の緯度データが返されます
lonlow: [戻り値]指定した低周波の経度データが返されます

◇輝度温度データ読込み　※左側の数字は行番号です

輝度温度は格納型が 2byte 符号なし整数(0～65535)で、スケール値が設定されているので、
輝度温度に付属しているアトリビュートにアクセスして、スケール値を読み出します。
スケールを適用する際は、欠損値(65535)と異常値(65534)は除外します。

```
386   // read array: tb for 06h
387   dhnd = H5Dopen(fhnd, "Brightness Temperature (6.9GHz,H)", H5P_DEFAULT);
388   ahnd = H5Aopen(dhnd, "SCALE FACTOR", H5P_DEFAULT); // get scale
389   ret = H5Aread(ahnd, H5T_NATIVE_FLOAT, &sca);       // get
390   ret = H5Aclose(ahnd);                               // get scale
391   ret = H5Dread(dhnd, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL, H5P_DEFAULT, tb06h);
392   if(ret < 0){
393     printf("H5Dread error: Brightness Temperature (6.9GHz,H)¥n");
394     exit(1);
395   }
396   ret = H5Dclose(dhnd);
397   // cutoff overlap & change scale
398   for(j = 0; j < num; ++j){
399     for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
400       tb06h[j][i] = tb06h[j+ovr][i];
401       if(tb06h[j][i] < 65534) tb06h[j][i] = tb06h[j][i] * sca;
402     }
403   }
404   for(j = num; j < LMT; ++j){
405     for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
406       tb06h[j][i] = 0;
407     }
408   }
409   // sample display
410   printf("tb06h[scan=0][pixel=0]: %9.2f¥n", tb06h[0][0]);
```

読込み

スケール読込み

重複除去

欠損値(65535)と異常値(65534)は除外して、
スケールを適用します。

このスケール処理は AMTK を使用する場合は
自動的に行われます。
→ P.16 基礎知識編３．１２参照

◇L1 品質フラグデータ読込み　※左側の数字は行番号です

L1 低周波品質フラグは 2 つの 1byte 整数型で 16 ビットの内 12 ビットが各低周波に対応します。
L1 高周波品質フラグは 1 つの 1byte 整数型で 8 ビットの内 4 ビットが各高周波に対応します。

読込み

```
802   // read array: pixel data quality for low
803   dhnd = H5Dopen(fhnd, "Pixel Data Quality 6 to 36", H5P_DEFAULT);
804   ret = H5Dread(dhnd, H5T_NATIVE_UCHAR, H5S_ALL, H5S_ALL, H5P_DEFAULT, i1d2hi);
805   if(ret < 0){
806     printf("H5Dread error: Pixel Data Quality 6 to 36¥n");
807     exit(1);
808   }
809   ret = H5Dclose(dhnd);
```

L1 品質フラグデータは、各周波数（6GHz、7GHz）、偏波に対する RFI 情報がまとまって取得されます。
まとまったままでは扱い難いので、各周波数毎の配列に分割します。
格納値は 2 ビットのマスク値なので、各周波数毎に unsigned char 型の 2 次元配列を用意します。
L1 品質フラグデータには RFI(Radio Frequency Interference; 電波干渉)情報が格納されています。
電波干渉の発生がない場合は 00、発生の可能性がある場合は 10、発生がある場合は 11 となります。

ビット毎の情報を各周波数に分解

重複除去も行います

```
810   // cutoff overlap & separate
811   for(j = 0; j < num; ++j){
812     for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
813       pdq06v[j][i]=0;
814       if((i1d2hi[j+ovr][i*2+0] &   1) != 0) pdq06v[j][i]=1;
815       if((i1d2hi[j+ovr][i*2+0] &   2) != 0) pdq06v[j][i]=10;
816       if(((i1d2hi[j+ovr][i*2+0] &   1) != 0) && ((i1d2hi[j+ovr][i*2+0] &   2) !=
0)) p  06v[j][i]=11;
817       pdq06h[j][i]=0;
818       if((i1d2hi[j+ovr][i*2+0] &   4) != 0) pdq06h[j][i]=1;
819       if((i1d2hi[j+ovr][i*2+0] &   8) != 0) pdq06h[j][i]=10;
820       if(((i1d2hi[j+ovr][i*2+0] &   4) != 0) && ((i1d2hi[j+ovr][i*2+0] &   8) !=
0)) p  06h[j][i]=11;
821       pdq07v[j][i]=0;
822       if((i1d2hi[j+ovr][i*2+0] &  16) != 0) pdq07v[j][i]=1;
          if((i1d2hi[j+ovr][i*2+0] &  32) != 0) pdq07v[j][i]=10;
          if(((i1d2hi[j+ovr][i*2+0] &  16) != 0) && ((i1d2hi[j+ovr][i*2+0] &  32) !=
  07v[j][i]=11;
          pdq07h[j][i]=0;
826       if((i1d2hi[j+ovr][i*2+0] &  64) != 0) pdq07h[j][i]=1;
          if((i1d2hi[j+ovr][i*2+0] & 128) != 0) pdq07h[j][i]=10;
          if(((i1d2hi[j+ovr][i*2+0] &  64) != 0) && ((i1d2hi[j+ovr][i*2+0] & 128) !=
  07h[j][i]=11;
829     }
830   }
831   for(j = num; j < LMT; ++j){
832     for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
833       pdq06h[j][i]=0;
834       pdq06v[j][i]=0;
835       pdq07h[j][i]=0;
836       pdq07v[j][i]=0;
837     }
838   }
```

◇L1B 陸海フラグデータ読込み　※左側の数字は行番号です

L1B 陸海は 1byte 整数型の 2 次元配列((スキャン数*チャンネル数)×ピクセル数)です。
低周波データは周波数が 6 チャンネル(6G/7G/10G/18G/23G/36G)あります。
高周波データは周波数が 2 チャンネル(89G A ホーン/89G B ホーン)あります。

P.14 基礎知識編３．９参照

読込み

```
876   // read array: land ocean flag for low
877   dhnd = H5Dopen(fhnd, "Land_Ocean Flag 6 to 36", H5P_DEFAULT);
878   ret = H5Dread(dhnd, H5T_NATIVE_UCHAR, H5S_ALL, H5S_ALL, H5P_DEFAULT, loflo);
      if(ret < 0){
        printf("H5Dread error: Land_Ocean Flag 6 to 36¥n");
881     exit(1);
882   }
883   ret = H5Dclose(dhnd);
```

陸海フラグデータは、解像度毎に全周波数データがまとまって取得されます。
まとまったままでは扱い難いので、各周波数毎の配列に分割します。
格納値は 0～100 のフラグ値なので、各周波数毎に 1byte 整数型の 2 次元配列を用意します。

各周波数に分解します

```
884   // separate
885   for(j = 0; j < num+ovr*2; ++j){
886     for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
887       lof06[j][i]=loflo[(num+ovr*2)*0+j][i];
888       lof07[j][i]=loflo[(num+ovr*2)*1+j][i];
889       lof10[j][i]=loflo[(num+ovr*2)*2+j][i];
          lof18[j][i]=loflo[(num+ovr*2)*3+j][i];
          lof23[j][i]=loflo[(num+ovr*2)*4+j][i];
          lof36[j][i]=loflo[(num+ovr*2)*5+j][i];
      }
894   }
```

重複除去も行います

```
      // cutoff overlap
      for(j = 0; j < num; ++j){
        for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
898       lof06[j][i]=lof06[j+ovr][i];
899       lof07[j][i]=lof07[j+ovr][i];
900       lof10[j][i]=lof10[j+ovr][i];
901       lof18[j][i]=lof18[j+ovr][i];
902       lof23[j][i]=lof23[j+ovr][i];
903       lof36[j][i]=lof36[j+ovr][i];
904     }
905   }
906   for(j = num; j < LMT; ++j){
907     for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
908       lof06[j][i]=0;
909       lof07[j][i]=0;
910       lof10[j][i]=0;
911       lof18[j][i]=0;
912       lof23[j][i]=0;
913       lof36[j][i]=0;
914     }
915   }
```

◇観測入射角データ読込み　※左側の数字は行番号です

観測入射角は格納型が 2byte 符号あり整数(-32768～32767)で、スケール値が設定されているので、
観測入射角に付属しているアトリビュートにアクセスして、スケール値を読み出します。
スケールを適用する際は、欠損値(-32768)と異常値(-32767)は除外します。

```
956  // read array: earth incidence
957  dhnd = H5Dopen(fhnd, "Earth Incidence", H5P_DEFAULT);
958  ahnd = H5Aopen(dhnd, "SCALE FACTOR", H5P_DEFAULT); // ge
959  ret = H5Aread(ahnd, H5T_NATIVE_FLOAT, &sca);       // ge
960  ret = H5Aclose(ahnd);                               // get scale
961  ret = H5Dread(dhnd, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL, H5P_DEFAULT,
);
962  if(ret < 0){
963    printf("H5Dread error: Earth Incidence¥n");
964    exit(1);
965  }
966  ret = H5Dclose(dhnd);
967  // cutoff overlap & change scale
968  for(j = 0; j < num; ++j){
969    for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
970      ear_in[j][i] = ear_in[j+ovr][i];
971      if(ear_in[j][i] > -32767) ear_in[j][i] = ear_in[j][i] * sca;
972    }
973  }
974  for(j = num; j < LMT; ++j){
975    for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
976      ear_in[j][i] = 0;
977    }
978  }
979  // sample display
980  printf("ear_in[scan=0][pixel=0]: %9.2f¥n",ear_in[0][0]);
```

スケール読込み

読込み

欠損値(-32768)と異常値(-32767)は除外して、スケールを適用します。

重複除去

◇終了処理　※左側の数字は行番号です

ファイルをクローズします。
HDF5 を終了します。

```
1008  // close
1009  ret = H5Fclose(fhnd);
1010  ret = H5close();
```

◇ファイルのクローズ
ret = H5Fclose(fhnd);
fhnd:ファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇HDF5 の終了
ret = H5close();
ret: [戻り値]失敗の場合は負の値

７．１．２ コンパイル方法(build_readL1B_hdf5_c.sh 解説)

コンパイルに使用するスクリプト build_readL1B_hdf5_c.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～8 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 8 SZIP=/home/user1/util/szip_2.1
 9
```

11 行目に、使用するコンパイラを指定します。
インテルコンパイラ(icc)または PG コンパイラ(pgcc)または GNU コンパイラ(gcc)を指定します。

```
10 # compiler
11 cc=icc
12
13 # source filename
14 csrc="readL1B_hdf5.c amsr2time_.c amsr2latlon_.c"
15
16 # output filename
17 out=readL1B_hdf5_c
18
19 # library order
20 lib="-lhdf5_hl -lhdf5 -lsz -lz -lm"
21
22 # c compile
23 cmd="$cc -g $csrc -o $out -I$HDF5/include -I$SZIP/include -L$HDF5/lib
-L$SZIP/lib $lib"
24 echo $cmd
25 $cmd
26
27 # garbage
28 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL1B_hdf5_c.sh
icc -g readL1B_hdf5.c amsr2time_.c amsr2latlon_.c -o readL1B_hdf5_c
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lhdf5_hl -lhdf5 -lsz -lz -lm
~
```

７．１．３ プログラム実行結果のサンプル

サンプルプログラムでは固定配列を多数使用しているため、環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞
$ unlimit

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、
4 つとも実行してください。
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited

```
$ ./readL1B_hdf5_c GW1AM2_201207261145_055A_L1SGBTBR_
0000000.h5
input file: GW1AM2_201207261145_055A_L1SGBTBR_000000
0.h5
GeophysicalName: Brightness Temperature
GranuleID: GW1AM2_201207261145_055A_L1SGBTBR_0000000
ObservationStartDateTime: 2012-07-26T11:45:43.018Z
EquatorCrossingDateTime: 2012-07-26T12:12:37.848Z
ObservationEndDateTime: 2012-07-26T12:35:09.735Z
NumberOfScans: 1979
limit of NumberOfScans = 2200
OverlapScans: 20
CoRegistrationParameterA1: 6G- 1.25000,7G- 1.00000,10
G- 1.25000,18G- 1.25000,23G- 1.25000,36G- 1.00000
CoRegistrationParameterA2: 6G- 0.00000,7G--0.10000,10
G--0.25000,18G- 0.00000,23G--0.25000,36G- 0.00000
amsr2time: AMSR2_LEAP_DATA = /export/emc3/util/common
/AMTK_AMSR2_DATA/leapsec.dat
amsr2time: year=1993 month= 7 tai93sec=   15638401.00
amsr2time: year=1994 month= 7 tai93sec=   47174402.00
amsr2time: year=1996 month= 1 tai93sec=   94608003.00
amsr2time: year=1997 month= 7 tai93sec=  141868804.00
amsr2time: year=1999 month= 1 tai93sec=  189302405.00
amsr2time: year=2006 month= 1 tai93sec=  410227206.00
amsr2time: year=2009 month= 1 tai93sec=  504921607.00
amsr2time: year=2012 month= 7 tai93sec=  615254408.00
amsr2time: number of leap second = 8
time[scan=0]: 2012/07/26 11:45:43
latlon89a[scan=0][pixel=0]: ( -73.3289, 136.7714)
latlon89b[scan=0][pixel=0]: ( -73.4038, 137.1498)
latlonIm[scan=0][pixel=0]: ( -73.3538, 136.6228)
latlon06[scan=0][pixel=0]: ( -73.3592, 136.6213)
latlon07[scan=0][pixel=0]: ( -73.3497, 136.6429)
latlon10[scan=0][pixel=0]: ( -73.3506, 136.6001)
latlon18[scan=0][pixel=0]: ( -73.3592, 136.6213)
latlon23[scan=0][pixel=0]: ( -73.3506, 136.6001)
latlon36[scan=0][pixel=0]: ( -73.3532, 136.6514)
tb06h[scan=0][pixel=0]:    173.28
tb06v[scan=0][pixel=0]:    208.22
tb07h[scan=0][pixel=0]:    173.07
tb07v[scan=0][pixel=0]:    207.54
tb10h[scan=0][pixel=0]:    170.94
tb10v[scan=0][pixel=0]:    204.95
```

```
tb18h[scan=0][pixel=0]:    164.85
tb18v[scan=0][pixel=0]:    199.84
tb23h[scan=0][pixel=0]:    163.22
tb23v[scan=0][pixel=0]:    196.53
tb36h[scan=0][pixel=0]:    156.56
tb36v[scan=0][pixel=0]:    186.39
tb89ah[scan=0][pixel=0]:    163.76
tb89av[scan=0][pixel=0]:    179.27
tb89bh[scan=0][pixel=0]:    170.60
tb89bv[scan=0][pixel=0]:    188.16
pdq06h[scan=0][pixel=0]):   0
pdq06v[scan=0][pixel=0]):   0
pdq07h[scan=0][pixel=0]):   0
pdq07v[scan=0][pixel=0]):   0
pdq10h[scan=0][pixel=0]):   0
pdq10v[scan=0][pixel=0]):   0
pdq18h[scan=0][pixel=0]):   0
pdq18v[scan=0][pixel=0]):   0
pdq23h[scan=0][pixel=0]):   0
pdq23v[scan=0][pixel=0]):   0
pdq36h[scan=0][pixel=0]):   0
pdq36v[scan=0][pixel=0]):   0
pdq89ah[scan=0][pixel=0]):   0
pdq89av[scan=0][pixel=0]):   0
pdq89ah[scan=0][pixel=0]):   0
pdq89av[scan=0][pixel=0]):   0
lof06[scan=0][pixel=0]: 100
lof07[scan=0][pixel=0]: 100
lof10[scan=0][pixel=0]: 100
lof18[scan=0][pixel=0]: 100
lof23[scan=0][pixel=0]: 100
lof36[scan=0][pixel=0]: 100
lof89a[scan=0][pixel=0]: 100
lof89b[scan=0][pixel=0]: 100
ear_in[scan=0][pixel=0]:     55.20
ear_az[scan=0][pixel=0]:    144.76
```

７．２ L1R データ読込み

７．２．１ サンプルプログラム readL1R_hdf5.c 解説

サンプルプログラム readL1R_hdf5.c では、L1R データファイルから、以下のメタデータと格納データを読込んで、89G A ホーン緯度経度から低周波用緯度経度を抽出し、内容をテキスト表示します。
このサンプルプログラムでは、格納されている L1R 輝度温度の一部のみ取扱います。L1R 輝度温度データ一覧については、P.14「３．８　L1 リサンプリングデータ」を参照ください。
TAI93 形式時刻の変換には、サブルーチンを用意しています。

メタデータ	格納データ
* GeophysicalName - GranuleID - ObservationStartDateTime - EquatorCrossingDateTime - ObservationEndDateTime * NumberOfScans - OverlapScans - CoRegistrationParameterA1 - CoRegistrationParameterA2	* Scan Time * Latitude of Observation Point for 89A - Latitude of Observation Point for 89B - Longitude of Observation Point for 89A - Longitude of Observation Point for 89B * Brightness Temperature (res06,6.9GHz,H) - Brightness Temperature (res06,6.9GHz,V) - Brightness Temperature (res06,7.3GHz,H) - Brightness Temperature (res06,7.3GHz,V) - Brightness Temperature (res10,10.7GHz,H) - Brightness Temperature (res10,10.7GHz,V) - Brightness Temperature (res23,18.7GHz,H) - Brightness Temperature (res23,18.7GHz,V) - Brightness Temperature (res23,23.8GHz,H) - Brightness Temperature (res23,23.8GHz,V) - Brightness Temperature (res36,36.5GHz,H) - Brightness Temperature (res36,36.5GHz,V) - Brightness Temperature (res36,89.0GHz,H) - Brightness Temperature (res36,89.0GHz,V) - Brightness Temperature (original,89GHz-A,H) - Brightness Temperature (original,89GHz-A,V) - Brightness Temperature (original,89GHz-B,H) - Brightness Temperature (original,89GHz-B,V) * Pixel Data Quality 6 to 36 - Pixel Data Quality 89 * Land_Ocean Flag 6 to 36 - Land_Ocean Flag 89 * Earth Incidence - Earth Azimuth
89G A ホーン緯度経度から抽出するデータ	
* 低周波用緯度経度 → P.15 基礎知識編３．１０参照	

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の HDF5 関数は、最初に使用する際に使用説明を記載します。

HDF5 ライブラリでは、ファイルをオープンした後に、各データについてもオープンとクローズを繰り返します。各データの読込みは以下のような流れになります。

データをオープンする →必要な場合はスケールを取得する
→ データを読込む → データをクローズする

◇変数宣言　※左側の数字は行番号です

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include "hdf5.h"
 4 #include "amsr2time.h"
 :

 6 // fixed value
 7 #define LMT 2200 // limit of NumberOfScans
 8 #define AM2_DEF_SNUM_HI 486 // high resolution pixel width
 9 #define AM2_DEF_SNUM_LO 243 // low  resolution pixel width
10
11 int main(int argc, char *argv[]){
12   // interface variable
13   int i,j;         // loop variable
14   herr_t ret;      // return status
15   char *buf = NULL; // text buffer
16   char *fn  = NULL; // filename
17   hid_t fhnd;      // file handle
18   hid_t ahnd;      // attribute handle
19   hid_t atyp;      // attribute type
20   hid_t dhnd;      // dataset handle
21   hid_t shnd;      // dataspace handle
22
23   // meta data
24   char *geo = NULL; // GeophysicalName
25   char *gid = NULL; // GranuleID
26   char *tm1 = NULL; // ObservationStartDateTime
27   char *tm2 = NULL; // EquatorCrossingDateTime
28   char *tm3 = NULL; // ObservationEndDateTime
29   int num;         // NumberOfScans
30   int ovr;         // OverlapScans
31   double prm1[7];   // CoRegistrationParameterA1
32   double prm2[7];   // CoRegistrationParameterA2

39   // array data
40   AM2_COMMON_SCANTIME st[LMT]; // scantime
41   float lat89ar[LMT][AM2_DEF_SNUM_HI]; // lat for 89a altitude revised
42   float lat89br[LMT][AM2_DEF_SNUM_HI]; // lat for 89b altitude revised
 :
44   float lon89ar[LMT][AM2_DEF_SNUM_HI]; // lon for 89a altitude revised
45   float lon89br[LMT][AM2_DEF_SNUM_HI]; // lon for 89b altitude revised
 :
47   float tb06h06 [LMT][AM2_DEF_SNUM_LO]; // tb for 06h, resolution 06G
48   float tb06v06 [LMT][AM2_DEF_SNUM_LO]; // tb for 06v, resolution 06G
 :
65   unsigned char pdq06h [LMT][AM2_DEF_SNUM_LO]; // pixel data quality for 06h
66   unsigned char pdq06v [LMT][AM2_DEF_SNUM_LO]; // pixel data quality for 06v
 :
74   unsigned char lof06 [LMT  ][AM2_DEF_SNUM_LO]; // land ocean flag for 06
75   unsigned char lof10 [LMT  ][AM2_DEF_SNUM_LO]; // land ocean flag for 10
 :
81   float ear_in[LMT][AM2_DEF_SNUM_LO]; // earth incidence
82   float ear_az[LMT][AM2_DEF_SNUM_LO]; // earth azimuth
```

HDF5 用ヘッダーファイルをインクルードします。

時刻変換サブルーチンヘッダーをインクルードします。

標準プロダクトのスキャン数上限は
2200 で充分ですが、
準リアルプロダクトを使用する場合は、
2 周回程度繋がる場合があるので、
LMT=9000 としてください。
(標準の 4 パス分程度)

HDF5 用インターフェイス
変数を宣言します。

物理量名
グラニュール ID
観測開始時刻
赤道通過時刻
観測終了時刻
スキャン数
オーバーラップスキャン数
相対レジストレーション係数 A1
相対レジストレーション係数 A2

メタデータ用変数
を宣言します。

時刻データには AM2_COMMON_SCANTIME 構造体を使います。
配列の次元数はデータによって異なります。
スキャン数はパスによって若干上下するので、上限を決めて宣言しておきます。
ここでは上限として LMT=2200 を設定しています。
AM2_DEF_SNUM_HI は高解像度ピクセル数(486)です。
AM2_DEF_SNUM_LO は低解像度ピクセル数(243)です。

時刻
緯度
経度
輝度温度
品質フラグ
陸海フラグ
観測入射角
観測方位角

配列データ用変数
を宣言します。

◇開始処理　※左側の数字は行番号です

HDF5 の初期化処理を行います。

◇HDF5 の初期化
ret = H5open();
ret: [戻り値]失敗の場合は負の値

```
 94   // hdf5 initialize
 95   ret = H5open();
 96   if(ret < 0){
 97     printf("h5open error: %d¥n" ,ret);
 98     exit(1);
 99   }
100
```

ファイルをオープンします。

◇HDF5 ファイルオープン
fhnd = H5Fopen(fn, label1, label2);
fn: オープンするファイル名を指定します
label1: アクセスモードを指定します H5F_ACC_RDONLY で読込専用になります
label2: H5P_DEFAULT を指定します
fhnd: [戻り値]開いたファイルのハンドル値、失敗の場合は負の値

```
101   // open
102   fhnd = H5Fopen(fn, H5F_ACC_RDONLY, H5P_DEFAULT);
103   if(fhnd < 0){
104     printf("H5Fopen error: %s¥n", fn);
105     exit(1);
106   }
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込むには、ファイルに付属しているアトリビュートを H5Aopen()でオープンします。
アトリビュートの型を調べてからデータを読込みます。
不要になったハンドルはクローズします。

◇アトリビュートのオープン
ahnd = H5Aopen(fhnd, nam, label);
fhnd: ファイルハンドル値を指定します
nam: アトリビュート名を指定します
label: H5P_DEFAULT を指定します
ahnd: [戻り値]アトリビュートのハンドル値
失敗の場合は負の値

◇アトリビュートタイプの取得
atyp = H5Aget_type(ahnd);
ahnd: アトリビュートハンドル値を指定します
atyp: [戻り値]アトリビュートタイプ値
失敗の場合は負の値

◇アトリビュート読込み
ret = H5Aread(ahnd, otyp, buf);
ahnd: アトリビュートハンドル値を指定します
otyp: 出力変数の型を指定します(読込んだデータは、出力変数の型へ自動変換されます)
buf: 出力変数を指定します
ret: [戻り値]失敗の場合は負の値

```
108   // read meta: GeophysicalName
109   ahnd = H5Aopen(fhnd, "GeophysicalName", H5P_DEFAULT);
110   atyp = H5Aget_type(ahnd);
111   ret = H5Aread(ahnd, atyp, &geo);
112   if(ret < 0){
113     printf("H5Aread error: GeophysicalName¥n");
114     exit(1);
115   }
116   ret = H5Aclose(ahnd);
117   printf("GeophysicalName: %s¥n", geo);
```

◇アトリビュートのクローズ
ret = H5Aclose(ahnd);
ahnd:アトリビュートハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

メタデータからスキャン数を読込みます。
配列データ読込みではスキャン数の指定が必要になります。

```
163   // read meta: NumberOfScans
164   ahnd = H5Aopen(fhnd, "NumberOfScans", H5P_DEFAULT);
165   atyp = H5Aget_type(ahnd);
166   ret = H5Aread(ahnd, atyp, &buf);
167   if(ret < 0){
168     printf("H5Aread error: NumberOfScans¥n");
169     exit(1);
170   }
171   ret = H5Aclose(ahnd);
172   num = atoi(buf);
173   printf("NumberOfScans: %d¥n", num);
```

メタデータは全て文字として取得されるので、
数値に変換しておきます。

◇時刻データ読込み　※左側の数字は行番号です

データを読込むためには、データセットを H5Dopen()でオープンします。
L1 データには前後にオーバーラップが含まれるので重複部分を取り除きます。
時刻データの格納形式は TAI93 形式なので年/月/日/時/分/秒に変換します。

P.11 基礎知識編３．５参照

◇データセットのオープン
dhnd = H5Dopen(fhnd, nam, label);
fhnd: ファイルハンドル値を指定します
nam: データセット名を指定します
label: H5P_DEFAULT を指定します
dhnd: [戻り値]データセットのハンドル値
失敗の場合は負の値

◇データ読込み
ret = H5Dread(dhnd, otyp, label1, label2, label3, buf);
dhnd: データセットハンドル値を指定します
otyp: 出力変数の型を指定します(読込んだデータは、出力変数の型へ自動変換されます)
label1,label2: 部分配列を読込む場合に使用します
全て読込む場合は、どちらも H5S_ALL を指定します
label3: H5P_DEFAULT を指定します
buf: 出力変数を指定します
ret: [戻り値]失敗の場合は負の値

◇データセットのクローズ
ret = H5Dclose(dhnd);
dhnd: データセットハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

```
252   // read array: scantime
253   dhnd = H5Dopen(fhnd, "Scan Time", H5P_DEFAULT);
254   ret = H5Dread(dhnd, H5T_NATIVE_DOUBLE, H5S_ALL, H5S_ALL, H5P_DEFAULT, r8d1);
255   if(ret < 0){
        printf("H5Dread error: Scan Time¥n");
        exit(1);
      }
      ret = H5Dclose(dhnd);
      // cutoff overlap
      for(j = 0; j < num; ++j){
        r8d1[j] = r8d1[j + ovr];
      }
      for(j = num; j < LMT; ++j){
        r8d1[j] = 0;
      }
      // convert
      amsr2time_(&num, r8d1, st);
      // sample display
      printf("time[scan=0]: %04d/%02d/%02d %02d:%02d:%02d¥n"
271   , st[0].year
272   , st[0].month
273   , st[0].day
274   , st[0].hour
275   , st[0].minute
276   , st[0].second
277   );
```

読込み

重複除去

時刻変換

AMSR2 プロダクトの時刻格納形式は、TAI93 形式と呼ばれる、うるう秒を含めた 1993/01/01 からの通算秒です。
このままでは日時情報として扱い難いので、
このサンプルプログラムでは年/月/日/時/分/秒に変換するサブルーチンを用意しています。この時刻変換は AMTK を使用する場合は自動的に行われます。
→ P.11 基礎知識編３．６参照

◆時刻形式の変換
amsr2time_(num,in,out)
num: スキャン数を指定します
in: TAI93 形式の時刻データを指定します
out: [戻り値]AM2_COMMON_SCANTIME 構造体で年/月/日/時/分/秒データが返されます

◇緯度経度データ読込み　※左側の数字は行番号です

89G 緯度経度は格納データを読込みます。
読込んだ後は、オーバーラップを取り除きます。
低周波用緯度経度は、89G A ホーン緯度経度の 1 から数えて奇数番目を抽出します。
※C 言語の場合、0 から数えて偶数番目になります。

P. 15 基礎知識編３． １０参照

緯度読込み

経度読込み

重複除去

低周波抽出

```
279   // read array: latlon for 89a altitude revised
280   // read lat
281   dhnd = H5Dopen(fhnd ,"Latitude of Observation Point for 89A", H5P_DEFAULT);
282   ret = H5Dread(dhnd,H5T_NATIVE_FLOAT,H5S_ALL, H5S_ALL, H5P_DEFAULT, lat89ar);
283   if(ret < 0){
        printf("H5Dread error: Latitude of Observation Point for 89A¥n");
        exit(1);
      }
287   ret = H5Dclose(dhnd);
288   // read lon
289   dhnd = H5Dopen(fhnd ,"Longitude of Observation Point for 89A", H5P_DEFAULT);
290   ret = H5Dread(dhnd,H5T_NATIVE_FLOAT,H5S_ALL, H5S_ALL, H5P_DEFAULT, lon89ar);
291   if(ret < 0){
        printf("H5Dread error: Longitude of Observation Point for 89A¥n");
        exit(1);
      }
295   ret = H5Dclose(dhnd);
296   // cutoff overlap
297   for(j = 0; j < num; ++j){
        for(i = 0; i < AM2_DEF_SNUM_HI; ++i){
299       lat89ar[j][i] = lat89ar[j+ovr][i];
300       lon89ar[j][i] = lon89ar[j+ovr][i];
        }
      }
303   for(j = num; j < LMT; ++j){
        for(i = 0; i < AM2_DEF_SNUM_HI; ++i){
          lat89ar[j][i] = 0;
          lon89ar[j][i] = 0;
        }
      }
  :

345   // read array: latlon for low resolution
346   for(j = 0; j < num; ++j){
        for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
          latlr[j][i] = lat89ar[j][i * 2];
349       lonlr[j][i] = lon89ar[j][i * 2];
        }
      }
```

◇輝度温度データ読込み　※左側の数字は行番号です

輝度温度は格納型が 2byte 符号なし整数(0～65535)で、スケール値が設定されているので、
輝度温度に付属しているアトリビュートにアクセスして、スケール値を読み出します。
スケールを適用する際は、欠損値(65535)と異常値(65534)は除外します。

```
354   // read array: tb for 06h, resolution 06G
355   dhnd = H5Dopen(fhnd, "Brightness Temperature (res06,6.9GHz,H)", H5P_DEFAULT);
356   ahnd = H5Aopen(dhnd, "SCALE FACTOR", H5P_DEFAULT); // get scale
357   ret = H5Aread(ahnd, H5T_NATIVE_FLOAT, &sca);       // get
358   ret = H5Aclose(ahnd);                              // get scale
359   ret = H5Dread(dhnd,H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL, H5P_DEFAULT, tb06h06);
360   if(ret < 0){
361     printf("H5Dread error: Brightness Temperature (res06,6.9GHz,H)¥n");
362     exit(1);
363   }
364   ret = H5Dclose(dhnd);
365   // cutoff overlap & change scale
366   for(j = 0; j < num; ++j){
367     for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
368       tb06h06[j][i] = tb06h06[j+ovr][i];
369       if(tb06h06[j][i] < 65534) tb06h06[j][i] = tb06h06[j][i] * sca;
370     }
371   }
372   for(j = num; j < LMT; ++j){
373     for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
374       tb06h06[j][i] = 0;
375     }
376   }
377   // sample display
378   printf("tb06h06[scan=0][pixel=0]: %9.2f¥n", tb06h06[0][0]);
```

読込み

スケール読込み

重複除去

欠損値(65535)と異常値(65534)は除外して、
スケールを適用します。

このスケール処理は AMTK を使用する場合は
自動的に行われます。
→ P.16 基礎知識編３．１２参照

◇L1 品質フラグデータ読込み　※左側の数字は行番号です

L1 低周波品質フラグは 2 つの 1byte 整数型で 16 ビットの内 12 ビットが各低周波に対応します。
L1 高周波品質フラグは 1 つの 1byte 整数型で 8 ビットの内 4 ビットが各高周波に対応します。

読込み

```
822   // read array: pixel data quality for low
823   dhnd = H5Dopen(fhnd, "Pixel Data Quality 6 to 36", H5P_DEFAULT);
824   ret = H5Dread(dhnd, H5T_NATIVE_UCHAR, H5S_ALL, H5S_ALL, H5P_DEFAULT, i1d2hi);
825   if(ret < 0){
826     printf("H5Dread error: Pixel Data Quality 6 to 36¥n");
827     exit(1);
828   }
829   ret = H5Dclose(dhnd);
```

L1 品質フラグデータは、各周波数（6GHz、7GHz）、偏波に対する RFI 情報がまとまって取得されます。
まとまったままでは扱い難いので、各周波数毎の配列に分割します。
格納値は 2 ビットのマスク値なので、各周波数毎に unsigned char 型の 2 次元配列を用意します。
L1 品質フラグデータには RFI(Radio Frequency Interference; 電波干渉)情報が格納されています。
電波干渉の発生がない場合は 00、発生の可能性がある場合は 10、発生がある場合は 11 となります。

ビット毎の情報を各周波数に分解

重複除去も行います

```
830   // cutoff overlap & separate
831   for(j = 0; j < num; ++j){
832     for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
833       pdq06v[j][i]=0;
834       if((i1d2hi[j+ovr][i*2+0] &   1) != 0) pdq06v[j][i]=1;
835       if((i1d2hi[j+ovr][i*2+0] &   2) != 0) pdq06v[j][i]=10;
836       if(((i1d2hi[j+ovr][i*2+0] &   1) != 0) && ((i1d2hi[j+ovr][i*2+0] &   2) !=
0))   q06v[j][i]=11;
837       pdq06h[j][i]=0;
838       if((i1d2hi[j+ovr][i*2+0] &   4) != 0) pdq06h[j][i]=1;
839       if((i1d2hi[j+ovr][i*2+0] &   8) != 0) pdq06h[j][i]=10;
840       if(((i1d2hi[j+ovr][i*2+0] &   4) != 0) && ((i1d2hi[j+ovr][i*2+0] &   8) !=
0))   q06h[j][i]=11;
841       pdq07v[j][i]=0;
842       if((i1d2hi[j+ovr][i*2+0] &  16) != 0) pdq07v[j][i]=1;
843       if((i1d2hi[j+ovr][i*2+0] &  32) != 0) pdq07v[j][i]=10;
          if(((i1d2hi[j+ovr][i*2+0] &  16) != 0) && ((i1d2hi[j+ovr][i*2+0] &  32) !=
q07v[j][i]=11;
          pdq07h[j][i]=0;
          if((i1d2hi[j+ovr][i*2+0] &  64) != 0) pdq07h[j][i]=1;
847       if((i1d2hi[j+ovr][i*2+0] & 128) != 0) pdq07h[j][i]=10;
          if(((i1d2hi[j+ovr][i*2+0] &  64) != 0) && ((i1d2hi[j+ovr][i*2+0] & 128) !=
q07h[j][i]=11;
        }
850   }
851   for(j = num; j < LMT; ++j){
852     for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
853       pdq06h[j][i]=0;
854       pdq06v[j][i]=0;
855       pdq07h[j][i]=0;
856       pdq07v[j][i]=0;
857     }
858   }
```

◇L1R 陸海フラグデータ読込み　※左側の数字は行番号です

L1R 陸海は 1byte 整数型の 2 次元配列((スキャン数*チャンネル数)×ピクセル数)です。
低周波データは解像度が 4 チャンネル(6G/10G/23G/36G)あります。
高周波データは解像度が 2 チャンネル(89G A ホーン/89G B ホーン)あります。

P.14 基礎知識編３．９参照

読込み

```
896   // read array: land ocean flag for low
897   dhnd = H5Dopen(fhnd, "Land_Ocean Flag 6 to 36", H5P_DEFAULT);
898   ret = H5Dread(dhnd, H5T_NATIVE_UCHAR, H5S_ALL, H5S_ALL, H5P_DEFAULT, loflo);
      if(ret < 0){
        printf("H5Dread error: Land_Ocean Flag 6 to 36¥n");
901     exit(1);
902   }
903   ret = H5Dclose(dhnd);
```

陸海フラグデータは、解像度毎に全周波数データがまとまって取得されます。
まとまったままでは扱い難いので、各周波数毎の配列に分割します。
格納値は 0～100 のフラグ値なので、各周波数毎に 1byte 整数型の 2 次元配列を用意します。

各周波数に分解します

```
904   // separate
905   for(j = 0; j < num+ovr*2; ++j){
906     for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
907       lof06[j][i]=loflo[(num+ovr*2)*0+j][i];
908       lof10[j][i]=loflo[(num+ovr*2)*1+j][i];
90        lof23[j][i]=loflo[(num+ovr*2)*2+j][i];
          lof36[j][i]=loflo[(num+ovr*2)*3+j][i];
      }
    }
    // cutoff overlap
914 for(j = 0; j < num; ++j){
```

重複除去も行います

```
      for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
        lof06[j][i]=lof06[j+ovr][i];
        lof10[j][i]=lof10[j+ovr][i];
918     lof23[j][i]=lof23[j+ovr][i];
919     lof36[j][i]=lof36[j+ovr][i];
920   }
921 }
922 for(j = num; j < LMT; ++j){
923   for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
924     lof06[j][i]=0;
925     lof10[j][i]=0;
926     lof23[j][i]=0;
927     lof36[j][i]=0;
928   }
929 }
```

◇観測入射角データ読込み　※左側の数字は行番号です

観測入射角は格納型が 2byte 符号あり整数(-32768～32767)で、スケール値が設定されているので、
観測入射角に付属しているアトリビュートにアクセスして、スケール値を読み出します。
スケールを適用する際は、欠損値(-32768)と異常値(-32767)は除外します。

```
968  // read array: earth incidence
969  dhnd = H5Dopen(fhnd, "Earth Incidence", H5P_DEFAULT);
970  ahnd = H5Aopen(dhnd, "SCALE FACTOR", H5P_DEFAULT); // ge
971  ret = H5Aread(ahnd, H5T_NATIVE_FLOAT, &sca);       // ge
972  ret = H5Aclose(ahnd);                               // get scale
973  ret = H5Dread(dhnd, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL, H5P_DEFAULT,
     );
974  if(ret < 0){
975    printf("H5Dread error: Earth Incidence¥n");
976    exit(1);
977  }
978  ret = H5Dclose(dhnd);
979  // cutoff overlap & change scale
980  for(j = 0; j < num; ++j){
981    for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
982      ear_in[j][i] = ear_in[j+ovr][i];
983      if(ear_in[j][i] > -32767) ear_in[j][i] = ear_in[j][i] * sca;
984    }
985  }
986  for(j = num; j < LMT; ++j){
987    for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
988      ear_in[j][i] = 0;
989    }
990  }
991  // sample display
992  printf("ear_in[scan=0][pixel=0]: %9.2f¥n",ear_in[0][0]);
```

スケール読込み

読込み

欠損値(-32768)と異常値(-32767)は除外して、スケールを適用します。

重複除去

◇終了処理　※左側の数字は行番号です

ファイルをクローズします。
HDF5 を終了します。

```
1020  // close
1021  ret = H5Fclose(fhnd);
1022  ret = H5close();
```

◇ファイルのクローズ
ret = H5Fclose(fhnd);
fhnd:ファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇HDF5 の終了
ret = H5close();
ret: [戻り値]失敗の場合は負の値

７．２．２ コンパイル方法(build_readL1R_hdf5_c.sh 解説)

コンパイルに使用するスクリプト build_readL1R_hdf5_c.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～8 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 8 SZIP=/home/user1/util/szip_2.1
 9
```

11 行目に、使用するコンパイラを指定します。
インテルコンパイラ(icc)または PG コンパイラ(pgcc)または GNU コンパイラ(gcc)を指定します。

```
10 # compiler
11 cc=icc
12
13 # source filename
14 csrc="readL1R_hdf5.c amsr2time_.c "
15
16 # output filename
17 out=readL1R_hdf5_c
18
19 # library order
20 lib="-lhdf5_hl -lhdf5 -lsz -lz -lm"
21
22 # c compile
23 cmd="$cc -g $csrc -o $out -I$HDF5/include -I$SZIP/include -L$HDF5/lib
-L$SZIP/lib $lib"
24 echo $cmd
25 $cmd
26
27 # garbage
28 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL1R_hdf5_c.sh
icc -g readL1R_hdf5.c amsr2time_.c -o readL1R_hdf5_c
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lhdf5_hl -lhdf5 -lsz -lz -lm
~
```

７．２．３ プログラム実行結果のサンプル

サンプルプログラムでは固定配列を多数使用しているため、環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞
$ unlimit

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、4 つとも実行してください。
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited

```
$ ./readL1R_hdf5_c GW1AM2_201207261145_055A_L1SGRTBR_
0000000.h5
input file: GW1AM2_201207261145_055A_L1SGRTBR_000000
0.h5
GeophysicalName: Brightness Temperature
GranuleID: GW1AM2_201207261145_055A_L1SGRTBR_0000000
ObservationStartDateTime: 2012-07-26T11:45:43.018Z
EquatorCrossingDateTime: 2012-07-26T12:12:37.848Z
ObservationEndDateTime: 2012-07-26T12:35:09.735Z
NumberOfScans: 1979
limit of NumberOfScans = 2200
OverlapScans: 20
CoRegistrationParameterA1: 6G- 0.00000,7G- 0.00000,10
G- 0.00000,18G- 0.00000,23G- 0.00000,36G- 0.00000
CoRegistrationParameterA2: 6G- 0.00000,7G- 0.00000,10
G- 0.00000,18G- 0.00000,23G- 0.00000,36G- 0.00000
amsr2time: AMSR2_LEAP_DATA = /export/emc3/util/common
/AMTK_AMSR2_DATA/leapsec.dat
amsr2time: year=1993 month= 7 tai93sec=   15638401.00
amsr2time: year=1994 month= 7 tai93sec=   47174402.00
amsr2time: year=1996 month= 1 tai93sec=   94608003.00
amsr2time: year=1997 month= 7 tai93sec=  141868804.00
amsr2time: year=1999 month= 1 tai93sec=  189302405.00
amsr2time: year=2006 month= 1 tai93sec=  410227206.00
amsr2time: year=2009 month= 1 tai93sec=  504921607.00
amsr2time: year=2012 month= 7 tai93sec=  615254408.00
amsr2time: number of leap second = 8
time[scan=0]: 2012/07/26 11:45:43
latlon89ar[scan=0][pixel=0]: ( -73.3581, 136.8432)
latlon89br[scan=0][pixel=0]: ( -73.4328, 137.2216)
latlonlr[scan=0][pixel=0]: ( -73.3581, 136.8432)
tb06h06[scan=0][pixel=0]:    173.41
tb06v06[scan=0][pixel=0]:    208.09
tb07h06[scan=0][pixel=0]:    173.07
tb07v06[scan=0][pixel=0]:    207.11
tb10h10[scan=0][pixel=0]:    170.40
tb10v10[scan=0][pixel=0]:    204.58
tb18h23[scan=0][pixel=0]:    165.83
tb18v23[scan=0][pixel=0]:    199.41
tb23h23[scan=0][pixel=0]:    163.55
tb23v23[scan=0][pixel=0]:    195.90
tb36h36[scan=0][pixel=0]:    153.55
tb36v36[scan=0][pixel=0]:    183.95
tb89h36[scan=0][pixel=0]:    163.86
tb89v36[scan=0][pixel=0]:    181.05
tb89ah[scan=0][pixel=0]:    163.76
tb89av[scan=0][pixel=0]:    179.27
tb89bh[scan=0][pixel=0]:    170.60
tb89bv[scan=0][pixel=0]:    188.16
pdq06h[scan=0][pixel=0]):   0
pdq06v[scan=0][pixel=0]):   0
pdq07h[scan=0][pixel=0]):   0
pdq07v[scan=0][pixel=0]):   0
pdq10h[scan=0][pixel=0]):   0
pdq10v[scan=0][pixel=0]):   0
pdq18h[scan=0][pixel=0]):   0
pdq18v[scan=0][pixel=0]):   0
pdq23h[scan=0][pixel=0]):   0
pdq23v[scan=0][pixel=0]):   0
pdq36h[scan=0][pixel=0]):   0
pdq36v[scan=0][pixel=0]):   0
pdq89ah[scan=0][pixel=0]):   0
pdq89av[scan=0][pixel=0]):   0
pdq89ah[scan=0][pixel=0]):   0
pdq89av[scan=0][pixel=0]):   0
lof06[scan=0][pixel=0]: 100
lof10[scan=0][pixel=0]: 100
lof23[scan=0][pixel=0]: 100
lof36[scan=0][pixel=0]: 100
lof89a[scan=0][pixel=0]: 100
lof89b[scan=0][pixel=0]: 100
ear_in[scan=0][pixel=0]:     55.20
ear_az[scan=0][pixel=0]:    144.76
```

７．３ L2 低解像度データ読込み

積算雲水量(CLW)・海氷密接度(SIC)・土壌水分量(SMC)・積雪深(SND)・海面水温(SST)・海上風速(SSW)・可降水量(TPW)が L2 低解像度です。降水量(PRC)は L2 高解像度を参照ください。

７．３．１ サンプルプログラム readL2L_hdf5.c 解説

サンプルプログラム readL2L_hdf5.c では、L2 低解像度データファイルから、以下のメタデータと格納データを読込んで、内容をテキスト表示します。
TAI93 形式時刻の変換には、サブルーチンを用意しています。

メタデータ	格納データ
* GeophysicalName - GranuleID - ObservationStartDateTime - EquatorCrossingDateTime - ObservationEndDateTime * NumberOfScans - OverlapScans	* Scan Time * Latitude of Observation Point * Longitude of Observation Point * Geophysical Data * Pixel Data Quality P.15 基礎知識編３．１０参照

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の HDF5 関数は、最初に使用する際に使用説明を記載します。

HDF5 ライブラリでは、ファイルをオープンした後に、各データについてもオープンとクローズを繰り返します。各データの読込みは以下のような流れになります。

データをオープンする →必要な場合はスケールを取得する
→ データを読込む → データをクローズする

◇変数宣言　※左側の数字は行番号です

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <string.h>
 4 #include "hdf5.h"
 5 #include "amsr2time.h"
 6
 7 // fixed value
 8 #define LMT 2200 // limit of NumberOfScans
 9 #define AM2_DEF_SNUM_HI 486 // high resolution pixel width
10 #define AM2_DEF_SNUM_LO 243 // low  resolution pixel width
 :
13 int main(int argc, char *argv[]){
14   // interface variable
15   int i,j;         // loop variable
16   herr_t ret;      // return status
17   char *buf = NULL; // text buffer
18   char *fn  = NULL; // filename
19   hid_t fhnd;      // file handle
20   hid_t ahnd;      // attribute handle
21   hid_t atyp;      // attribute type
22   hid_t dhnd;      // dataset handle
23
24   // meta data
25   char *geo = NULL; // GeophysicalName
26   char *gid = NULL; // GranuleID
27   char *tm1 = NULL; // ObservationStartDateTime
28   char *tm2 = NULL; // EquatorCrossingDateTime
29   char *tm3 = NULL; // ObservationEndDateTime
30   int num;         // NumberOfScans
31   int ovr;         // OverlapScans

37   // array data
38   AM2_COMMON_SCANTIME st[LMT]; // scantime
39   float lat[LMT][AM2_DEF_SNUM_LO]; // lat
40   float lon[LMT][AM2_DEF_SNUM_LO]; // lon
41   float geo1[LMT][AM2_DEF_SNUM_LO]; // geophysical data layer 1
42   float geo2[LMT][AM2_DEF_SNUM_LO]; // geophysical data layer 2
43   float geotmp[LMT*2][AM2_DEF_SNUM_LO]; // geophysical data temporary

44   unsigned char pdq1[LMT][AM2_DEF_SNUM_LO]; // pixel data quality layer 1
45   unsigned char pdq2[LMT][AM2_DEF_SNUM_LO]; // pixel data quality layer 2
46   unsigned char pdqtmp[LMT*2][AM2_DEF_SNUM_LO]; // pixel data quality temporary
```

HDF5 用ヘッダーファイルをインクルードします。

時刻変換サブルーチンヘッダーをインクルードします。

標準プロダクトのスキャン数上限は
2200 で充分ですが、
準リアルプロダクトを使用する場合は、
2 周回程度繋がる場合があるので、
LMT=9000 としてください。
(標準の 4 パス分程度)

HDF5 用インターフェイス
変数を宣言します。

物理量名
グラニュール ID
観測開始時刻
赤道通過時刻
観測終了時刻
スキャン数
オーバーラップスキャン数

メタデータ用変数
を宣言します。

時刻データには AM2_COMMON_SCANTIME 構造体を使います。
配列の次元数はデータによって異なります。
スキャン数はパスによって若干上下するので、上限を決めて宣言しておきます。
ここでは上限として LMT=2200 を設定しています。
AM2_DEF_SNUM_HI は高解像度ピクセル数(486)です。
AM2_DEF_SNUM_LO は低解像度ピクセル数(243)です。

時刻
緯度
経度
物理量
品質フラグ

配列データ用変数
を宣言します。

◇開始処理　※左側の数字は行番号です

HDF5 の初期化処理を行います。

◇HDF5 の初期化
ret = H5open();
ret: [戻り値]失敗の場合は負の値

```
58   // hdf5 initialize
59   ret = H5open();
60   if(ret < 0){
61     printf("h5open error: %d¥n" ,ret);
62     exit(1);
63   }
```

ファイルをオープンします。

◇HDF5 ファイルオープン
fhnd = H5Fopen(fn, label1, label2);
fn: オープンするファイル名を指定します
label1: アクセスモードを指定します H5F_ACC_RDONLY で読込専用になります
label2: H5P_DEFAULT を指定します
fhnd: [戻り値]開いたファイルのハンドル値、失敗の場合は負の値

```
65   // open
66   fhnd = H5Fopen(fn, H5F_ACC_RDONLY, H5P_DEFAULT);
67   if(fhnd < 0){
68     printf("H5Fopen error: %s¥n", fn);
69     exit(1);
70   }
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込むには、ファイルに付属しているアトリビュートを H5Aopen()でオープンします。
アトリビュートの型を調べてからデータを読込みます。
不要になったハンドルはクローズします。

◇アトリビュートのオープン
ahnd = H5Aopen(fhnd, nam, label);
fhnd: ファイルハンドル値を指定します
nam: アトリビュート名を指定します
label: H5P_DEFAULT を指定します
ahnd: [戻り値]アトリビュートのハンドル値
失敗の場合は負の値

◇アトリビュートタイプの取得
atyp = H5Aget_type(ahnd);
ahnd: アトリビュートハンドル値を指定します
atyp: [戻り値]アトリビュートタイプ値
失敗の場合は負の値

◇アトリビュート読込み
ret = H5Aread(ahnd, otyp, buf);
ahnd: アトリビュートハンドル値を指定します
otyp: 出力変数の型を指定します(読込んだデータは、出力変数の型へ自動変換されます)
buf: 出力変数を指定します
ret: [戻り値]失敗の場合は負の値

```
72   // read meta: GeophysicalName
73   ahnd = H5Aopen(fhnd, "GeophysicalName", H5P_DEFAULT);
74   atyp = H5Aget_type(ahnd);
75   ret = H5Aread(ahnd, atyp, &geo);
76   if(ret < 0){
77     printf("H5Aread error: GeophysicalName¥n");
78     exit(1);
79   }
80   ret = H5Aclose(ahnd);
81   printf("GeophysicalName: %s¥n", geo);
```

◇アトリビュートのクローズ
ret = H5Aclose(ahnd);
ahnd:アトリビュートハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

メタデータからスキャン数を読込みます。
配列データ読込みではスキャン数の指定が必要になります。

```
140   // read meta: NumberOfScans
141   ahnd = H5Aopen(fhnd, "NumberOfScans", H5P_DEFAULT);
142   atyp = H5Aget_type(ahnd);
143   ret = H5Aread(ahnd, atyp, &buf);
144   if(ret < 0){
145     printf("H5Aread error: NumberOfScans¥n");
146     exit(1);
147   }
148   ret = H5Aclose(ahnd);
149   num = atoi(buf);
150   printf("NumberOfScans: %d¥n", num);
```

メタデータは全て文字として取得されるので、
数値に変換しておきます。

◇時刻データ読込み　※左側の数字は行番号です

データを読込むためには、データセットを H5Dopen()でオープンします。
時刻データの格納形式は TAI93 形式なので年/月/日/時/分/秒に変換します。

◇データセットのオープン
dhnd = H5Dopen(fhnd, nam, label);
fhnd: ファイルハンドル値を指定します
nam: データセット名を指定します
label: H5P_DEFAULT を指定します
dhnd: [戻り値]データセットのハンドル値
失敗の場合は負の値

◇データ読込み
ret = H5Dread(dhnd, otyp, label1, label2, label3, buf);
dhnd: データセットハンドル値を指定します
otyp: 出力変数の型を指定します(読込んだデータは、出力変数の型へ自動変換されます)
label1,label2: 部分配列を読込む場合に使用します
全て読込む場合は、どちらも H5S_ALL を指定します
label3: H5P_DEFAULT を指定します
buf: 出力変数を指定します
ret: [戻り値]失敗の場合は負の値

◇データセットのクローズ
ret = H5Dclose(dhnd);
dhnd: データセットハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

```
173   // read array: scantime
174   dhnd = H5Dopen(fhnd, "Scan Time", H5P_DEFAULT);
175   ret = H5Dread(dhnd, H5T_NATIVE_DOUBLE, H5S_ALL, H5S_ALL, H5P_DEFAULT, r8d1);
176   if(ret < 0){
        printf("H5Dread error: Scan Time¥n");
        exit(1);
      }
180   ret = H5Dclose(dhnd);
181   // convert
      amsr2time_(&num, r8d1, st);
      // sample display
184   printf("time[scan=0]: %04d/%02d/%02d %02d:%02d:%02d¥n"
185   , st[0].year
186   , st[0].month
187   , st[0].day
188   , st[0].hour
189   , st[0].minute
190   , st[0].second
191   );
```

読込み

時刻変換

AMSR2 プロダクトの時刻格納形式は、TAI93 形式と呼ばれる、うるう秒を含めた 1993/01/01 からの通算秒です。
このままでは日時情報として扱い難いので、
このサンプルプログラムでは年/月/日/時/分/秒に変換するサブルーチンを用意しています。この時刻変換は AMTK を使用する場合は自動的に行われます。
→ P.11 基礎知識編３．６参照

◆時刻形式の変換
amsr2time_(num,in,out)
num: スキャン数を指定します
in: TAI93 形式の時刻データを指定します
out: [戻り値]AM2_COMMON_SCANTIME 構造体で年/月/日/時/分/秒データが返されます

◇緯度経度データ読込み　※左側の数字は行番号です

緯度経度データを読込みます。 → P.15 基礎知識編３．１０参照

緯度読込み

経度読込み

```
200   // read array: latlon
201   // read lat
202   dhnd = H5Dopen(fhnd ,"Latitude of Observation Point", H5P_DEFAULT);
203   ret = H5Dread(dhnd, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL, H5P_DEFAULT, lat);
204   if(ret < 0){
        printf("H5Dread error: Latitude of Observation Point¥n");
        exit(1);
207   }
208   ret = H5Dclose(dhnd);
209   // read lon
210   dhnd = H5Dopen(fhnd ,"Longitude of Observation Point", H5P_DEFAULT);
211   ret = H5Dread(dhnd, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL, H5P_DEFAULT, lon);
212   if(ret < 0){
        printf("H5Dread error: Longitude of Observation Point¥n");
        exit(1);
215   }
216   ret = H5Dclose(dhnd);
217   // sample display
218   printf("latlon[scan=0][pixel=0]: (%9.4f,%9.4f)¥n", lat[0][0], lon[0][0]);
```

◇物理量データ読込み　※左側の数字は行番号です

物理量データを読込みます。
積雪深(SND)・海面水温(SST)は物理量が 2 層あるので、この 2 つとそれ以外で処理を分けています。
積雪深の 2 層目は積雪水量[cm]です。海面水温の 2 層目は 10GHz による SST[℃]です。

物理量は格納型が 2byte 符号あり整数(-32768～32767)で、スケール値が設定されているので、
物理量に付属しているアトリビュートにアクセスして、スケール値を読み出します。
スケールを適用する際は、欠損値(-32768)と異常値(-32767)は除外します。

```
213   // read array: geophysical data for 1 layer
214   if(strncmp(gid+29,"SND",3)!=0 && strncmp(gid+29,"SST",3)!=0 ){
215     dhnd = H5Dopen(fhnd, "Geophysical Data", H5P_DEFAULT);
216     // get scale
217     ahnd = H5Aopen(dhnd, "SCALE FACTOR", H5P_DEFAULT);
218     ret = H5Aread(ahnd, H5T_NATIVE_FLOAT, &sca);
219     ret = H5Aclose(ahnd);
220     // read
221     ret = H5Dread(dhnd, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL, H5P_DEFAULT, geo1);
222     if(ret < 0){
223       printf("H5Dread error: Geophysical Data¥n");
224       exit(1);
225     }
226     ret = H5Dclose(dhnd);
227     // change scale
228     for(j = 0; j < num; ++j){
229       for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
230         if(geo1[j][i] > -32767) geo1[j][i] = geo1[j][i] * sca;
231       }
232     }
233   }
234
```

スケール読込み

読込み

スケール処理

このスケール処理は AMTK を使用する場合は
自動的に行われます。
→ P.16 基礎知識編３．１２参照

欠損値(-32768)と異常値(-32767)は除外して、
スケールを適用します。

2 層ある場合は、一時変数で全体を読込んだ後に、層ごとに分割しています。

```
235   // read array: geophysical data for 2 layer
236   if(strncmp(gid+29,"SND",3)==0 || strncmp(gid+29,"SST",3)==0 ){
237     dhnd = H5Dopen(fhnd, "Geophysical Data", H5P_DEFAULT);
238     // get scale
239     ahnd = H5Aopen(dhnd, "SCALE FACTOR", H5P_DEFAULT);
240     ret = H5Aread(ahnd, H5T_NATIVE_FLOAT, &sca);
241     ret = H5Aclose(ahnd);
242     // read
243     ret = H5Dread(dhnd, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL, H5P_DEFAULT,
geotmp);
244     if(ret < 0){
245       printf("H5Dread error: Geophysical Data¥n");
246       exit(1);
247     }
248     ret = H5Dclose(dhnd);
249     // separate & change scale
250     for(j = 0; j < num; ++j){
251       for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
252         geo1[j][i] = geotmp[j*2][i*2+0];
253         geo2[j][i] = geotmp[j*2][i*2+1];
254         if(geo1[j][i] > -32767) geo1[j][i] = geo1[j][i] * sca;
255         if(geo2[j][i] > -32767) geo2[j][i] = geo2[j][i] * sca;
256       }
257     }
258   }
```

◇L2 品質フラグデータ読込み　※左側の数字は行番号です

L2 品質フラグデータを読込みます。
積雪深(SND)、海面水温(SST)は品質フラグが 2 層あるのでこの 2 つとそれ以外で処理を分けています。

```
260   // read array: pixel data quality for 1 layer
261   if(strncmp(gid+29,"SND",3)!=0 && strncmp(gid+29,"SST",3)!=0 ){
262     dhnd = H5Dopen(fhnd, "Pixel Data Quality", H5P_DEFAULT);
263     ret = H5Dread(dhnd, H5T_NATIVE_UCHAR, H5S_ALL, H5S_ALL, H5P_DEFAULT, pdq1);
264     if(ret < 0){
265       printf("H5Dread error: Pixel Data Quality¥n");
266       exit(1);
267     }
268     ret = H5Dclose(dhnd);
269   }
270
```

2 層ある場合は、一時変数で全体を読込んだ後に、層ごとに分割しています。

```
271   // read array: pixel data quality for 2 layer
272   if(strncmp(gid+29,"SND",3)==0 || strncmp(gid+29,"SST",3)==0 ){
273     // read
274     dhnd = H5Dopen(fhnd, "Pixel Data Quality", H5P_DEFAULT);
275     ret = H5Dread(dhnd, H5T_NATIVE_UCHAR, H5S_ALL, H5S_ALL, H5P_DEFAULT,
pdqtmp);
276     if(ret < 0){
277       printf("H5Dread error: Pixel Data Quality¥n");
278       exit(1);
279     }
280     ret = H5Dclose(dhnd);
281     // separate
282     for(j = 0; j < num; ++j){
283       for(i = 0; i < AM2_DEF_SNUM_LO; ++i){
284         pdq1[j][i] = pdqtmp[num*0+j][i];
285         pdq2[j][i] = pdqtmp[num*1+j][i];
286       }
287     }
288   }
```

L2 品質フラグには、アルゴリズム開発者が設定した物理量算出に係る補足情報が格納されています。
0～15 は OK 状態、16～255 は NG 状態を現します。
NG 状態の場合、物理量には欠損値(-32768)または異常値(-32767)が格納されています。
L2 品質フラグの詳細については、「AMSR2 高次プロダクトフォーマット説明書」(*1)を参照ください。
(*1)http://suzaku.eorc.jaxa.jp/GCOM_W/data/data_w_format_j.html

◇終了処理　※左側の数字は行番号です

ファイルをクローズします。
HDF5 を終了します。

```
314   // close
315   ret = H5Fclose(fhnd);
316   ret = H5close();
```

◇ファイルのクローズ
ret = H5Fclose(fhnd);
fhnd:ファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇HDF5 の終了
ret = H5close();
ret: [戻り値]失敗の場合は負の値

７．３．２ コンパイル方法(build_readL2L_hdf5_c.sh 解説)

コンパイルに使用するスクリプト build_readL2L_hdf5_c.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～8 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 8 SZIP=/home/user1/util/szip_2.1
 9
```

11 行目に、使用するコンパイラを指定します。
インテルコンパイラ(icc)または PG コンパイラ(pgcc)または GNU コンパイラ(gcc)を指定します。

```
10 # compiler
11 cc=icc
12
13 # source filename
14 csrc="readL2L_hdf5.c amsr2time_.c "
15
16 # output filename
17 out=readL2L_hdf5_c
18
19 # library order
20 lib="-lhdf5_hl -lhdf5 -lsz -lz -lm"
21
22 # c compile
23 cmd="$cc -g $csrc -o $out -I$HDF5/include -I$SZIP/include -L$HDF5/lib
-L$SZIP/lib $lib"
24 echo $cmd
25 $cmd
26
27 # garbage
28 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL2L_hdf5_c.sh
icc -g readL2L_hdf5.c amsr2time_.c -o readL2L_hdf5_c
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lhdf5_hl -lhdf5 -lsz -lz -lm
~
```

７．３．３ プログラム実行結果のサンプル

サンプルプログラムでは固定配列を多数使用しているため、環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞

```
$ unlimit
```

＜sh/bash 環境の場合＞

※以下のコマンドを順番に、4 つとも実行してください。

```
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited
```

```
$ ./readL2L_hdf5_c GW1AM2_201303011809_125D_L2SGCLWLA0000000.h5
input file: GW1AM2_201303011809_125D_L2SGCLWLA0000000.h5
GeophysicalName: Cloud Liquid Water
GranuleID: GW1AM2_201303011809_125D_L2SGCLWLA0000000
ObservationStartDateTime: 2013-03-01T18:09:10.122Z
EquatorCrossingDateTime: 2013-03-01T18:35:54.849Z
ObservationEndDateTime: 2013-03-01T18:58:26.342Z
NumberOfScans: 1972
limit of NumberOfScans = 2200
OverlapScans: 0
amsr2time: AMSR2_LEAP_DATA = /export/emc3/util/common/AMTK_AMSR2_DATA/leapsec.dat
amsr2time: year=1993 month= 7 tai93sec=   15638401.00
amsr2time: year=1994 month= 7 tai93sec=   47174402.00
amsr2time: year=1996 month= 1 tai93sec=   94608003.00
amsr2time: year=1997 month= 7 tai93sec=  141868804.00
amsr2time: year=1999 month= 1 tai93sec=  189302405.00
amsr2time: year=2006 month= 1 tai93sec=  410227206.00
amsr2time: year=2009 month= 1 tai93sec=  504921607.00
amsr2time: year=2012 month= 7 tai93sec=  615254408.00
amsr2time: number of leap second = 8
time[scan=0]: 2013/03/01 18:09:10
latlon[scan=0][pixel=0]: (  84.4574, -78.1076)
geo1[scan=0][pixel=0]: -32767.000 [Kg/m2] (PDQ:112)
```

７．４ L2 高解像度データ読込み

降水量(PRC)が L2 高解像度です。
積算雲水量(CLW)・海氷密接度(SIC)・土壌水分量(SMC)・積雪深(SND)・海面水温(SST)・海上風速(SSW)・可降水量(TPW)は L2 低解像度を参照ください。

７．４．１ サンプルプログラム readL2H_hdf5.c 解説

サンプルプログラム readL2H_hdf5.c では、L2 高解像度データファイルから、以下のメタデータと格納データを読込んで、内容をテキスト表示します。
TAI93 形式時刻の変換には、サブルーチンを用意しています。

メタデータ	格納データ
* GeophysicalName - GranuleID - ObservationStartDateTime - EquatorCrossingDateTime - ObservationEndDateTime * NumberOfScans - OverlapScans	* Scan Time * Latitude of Observation Point for 89A - Latitude of Observation Point for 89B * Longitude of Observation Point for 89A - Longitude of Observation Point for 89B * Geophysical Data for 89A - Geophysical Data for 89B * Pixel Data Quality for 89A - Pixel Data Quality for 89B P.15 基礎知識編３．１０参照

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の HDF5 関数は、最初に使用する際に使用説明を記載します。

HDF5 ライブラリでは、ファイルをオープンした後に、各データについてもオープンとクローズを繰り返します。各データの読込みは以下のような流れになります。

データをオープンする →必要な場合はスケールを取得する
→ データを読込む → データをクローズする

◇変数宣言　※左側の数字は行番号です

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <string.h>
 4 #include "hdf5.h"
 5 #include "amsr2time.h"
 6
 7 // fixed value
 8 #define LMT 2200 // limit of NumberOfScans
 9 #define AM2_DEF_SNUM_HI 486 // high resolution pixel width
10 #define AM2_DEF_SNUM_LO 243 // low  resolution pixel width
 :
13 int main(int argc, char *argv[]){
14   // interface variable
15   int i,j;         // loop variable
16   herr_t ret;      // return status
17   char *buf = NULL; // text buffer
18   char *fn  = NULL; // filename
19   hid_t fhnd;      // file handle
20   hid_t ahnd;      // attribute handle
21   hid_t atyp;      // attribute type
22   hid_t dhnd;      // dataset handle
23
24   // meta data
25   char *geo = NULL; // GeophysicalName
26   char *gid = NULL; // GranuleID
27   char *tm1 = NULL; // ObservationStartDateTime
28   char *tm2 = NULL; // EquatorCrossingDateTime
29   char *tm3 = NULL; // ObservationEndDateTime
30   int num;         // NumberOfScans
31   int ovr;         // OverlapScans

37   // array data
38   AM2_COMMON_SCANTIME st[LMT]; // scantime  時刻
39   float lat89a[LMT][AM2_DEF_SNUM_HI]; // lat for 89a  緯度
40   float lat89b[LMT][AM2_DEF_SNUM_HI]; // lat for 89b
41   float lon89a[LMT][AM2_DEF_SNUM_HI]; // lon for 89a  経度
42   float lon89b[LMT][AM2_DEF_SNUM_HI]; // lon for 89b
43   float geo1_89a[LMT][AM2_DEF_SNUM_HI];                物理量
44   float geo1_89b[LMT][AM2_DEF_SNUM_HI];
45   unsigned char pdq1_89a[LMT][AM2_DEF_SNUM_HI];        品質フラグ
46   unsigned char pdq1_89b[LMT][AM2_DEF_SNUM_HI];
```

HDF5 用ヘッダーファイルをインクルードします。

時刻変換サブルーチンヘッダーをインクルードします。

標準プロダクトのスキャン数上限は
2200 で充分ですが、
準リアルプロダクトを使用する場合は、
2 周回程度繋がる場合があるので、
LMT=9000 としてください。
(標準の 4 パス分程度)

HDF5 用インターフェイス
変数を宣言します。

物理量名
グラニュール ID
観測開始時刻
赤道通過時刻
観測終了時刻
スキャン数
オーバーラップスキャン数

メタデータ用変数
を宣言します。

時刻データには AM2_COMMON_SCANTIME 構造体を使います。
配列の次元数はデータによって異なります。
スキャン数はパスによって若干上下するので、上限を決めて宣言しておきます。
ここでは上限として LMT=2200 を設定しています。
AM2_DEF_SNUM_HI は高解像度ピクセル数(486)です。
AM2_DEF_SNUM_LO は低解像度ピクセル数(243)です。

配列データ用変数
を宣言します。

◇開始処理　※左側の数字は行番号です

HDF5 の初期化処理を行います。

◇HDF5 の初期化
ret = H5open();
ret: [戻り値]失敗の場合は負の値

```
58   // hdf5 initialize
59   ret = H5open();
60   if(ret < 0){
61     printf("h5open error: %d¥n" ,ret);
62     exit(1);
63   }
```

ファイルをオープンします。

◇HDF5 ファイルオープン
fhnd = H5Fopen(fn, label1, label2);
fn: オープンするファイル名を指定します
label1: アクセスモードを指定します H5F_ACC_RDONLY で読込専用になります
label2: H5P_DEFAULT を指定します
fhnd: [戻り値]開いたファイルのハンドル値、失敗の場合は負の値

```
65   // open
66   fhnd = H5Fopen(fn, H5F_ACC_RDONLY, H5P_DEFAULT);
67   if(fhnd < 0){
68     printf("H5Fopen error: %s¥n", fn);
69     exit(1);
70   }
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込むには、ファイルに付属しているアトリビュートを H5Aopen()でオープンします。
アトリビュートの型を調べてからデータを読込みます。
不要になったハンドルはクローズします。

◇アトリビュートのオープン
ahnd = H5Aopen(fhnd, nam, label);
fhnd: ファイルハンドル値を指定します
nam: アトリビュート名を指定します
label: H5P_DEFAULT を指定します
ahnd: [戻り値]アトリビュートのハンドル値
失敗の場合は負の値

◇アトリビュートタイプの取得
atyp = H5Aget_type(ahnd);
ahnd: アトリビュートハンドル値を指定します
atyp: [戻り値]アトリビュートタイプ値
失敗の場合は負の値

◇アトリビュート読込み
ret = H5Aread(ahnd, otyp, buf);
ahnd: アトリビュートハンドル値を指定します
otyp: 出力変数の型を指定します(読込んだデータは、出力変数の型へ自動変換されます)
buf: 出力変数を指定します
ret: [戻り値]失敗の場合は負の値

```
72   // read meta: GeophysicalName
73   ahnd = H5Aopen(fhnd, "GeophysicalName", H5P_DEFAULT);
74   atyp = H5Aget_type(ahnd);
75   ret = H5Aread(ahnd, atyp, &geo);
76   if(ret < 0){
77     printf("H5Aread error: GeophysicalName¥n");
78     exit(1);
79   }
80   ret = H5Aclose(ahnd);
81   printf("GeophysicalName: %s¥n", geo);
```

◇アトリビュートのクローズ
ret = H5Aclose(ahnd);
ahnd:アトリビュートハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

メタデータからスキャン数を読込みます。
配列データ読込みではスキャン数の指定が必要になります。

```
133   // read meta: NumberOfScans
134   ahnd = H5Aopen(fhnd, "NumberOfScans", H5P_DEFAULT);
135   atyp = H5Aget_type(ahnd);
136   ret = H5Aread(ahnd, atyp, &buf);
137   if(ret < 0){
138     printf("H5Aread error: NumberOfScans¥n");
139     exit(1);
140   }
141   ret = H5Aclose(ahnd);
142   num = atoi(buf);
143   printf("NumberOfScans: %d¥n", num);
```

メタデータは全て文字として取得されるので、
数値に変換しておきます。

◇時刻データ読込み　※左側の数字は行番号です

データを読込むためには、データセットを H5Dopen()でオープンします。
時刻データの格納形式は TAI93 形式なので年/月/日/時/分/秒に変換します。

◇データセットのオープン
dhnd = H5Dopen(fhnd, nam, label);
fhnd: ファイルハンドル値を指定します
nam: データセット名を指定します
label: H5P_DEFAULT を指定します
dhnd: [戻り値]データセットのハンドル値
失敗の場合は負の値

◇データ読込み
ret = H5Dread(dhnd, otyp, label1, label2, label3, buf);
dhnd: データセットハンドル値を指定します
otyp: 出力変数の型を指定します(読込んだデータは、出力変数の型へ自動変換されます)
label1,label2: 部分配列を読込む場合に使用します
全て読込む場合は、どちらも H5S_ALL を指定します
label3: H5P_DEFAULT を指定します
buf: 出力変数を指定します
ret: [戻り値]失敗の場合は負の値

◇データセットのクローズ
ret = H5Dclose(dhnd);
dhnd: データセットハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

読込み

時刻変換

```
166   // read array: scantime
167   dhnd = H5Dopen(fhnd, "Scan Time", H5P_DEFAULT);
168   ret = H5Dread(dhnd, H5T_NATIVE_DOUBLE, H5S_ALL, H5S_ALL, H5P_DEFAULT, r8d1);
169   if(ret < 0){
        printf("H5Dread error: Scan Time¥n");
        exit(1);
      }
173   ret = H5Dclose(dhnd);
174   // convert
      amsr2time_(&num, r8d1, st);
      // sample display
177   printf("time[scan=0]: %04d/%02d/%02d %02d:%02d:%02d¥n"
178   , st[0].year
179   , st[0].month
180   , st[0].day
181   , st[0].hour
182   , st[0].minute
183   , st[0].second
184   );
```

AMSR2 プロダクトの時刻格納形式は、TAI93 形式と呼ばれる、うるう秒を含めた 1993/01/01 からの通算秒です。
このままでは日時情報として扱い難いので、
このサンプルプログラムでは年/月/日/時/分/秒に変換するサブルーチンを用意しています。この時刻変換は AMTK を使用する場合は自動的に行われます。
→ P.11 基礎知識編３．６参照

◆時刻形式の変換
amsr2time_(num,in,out)
num: スキャン数を指定します
in: TAI93 形式の時刻データを指定します
out: [戻り値]AM2_COMMON_SCANTIME 構造体で年/月/日/時/分/秒データが返されます

◇緯度経度データ読込み　※左側の数字は行番号です

緯度経度データを読込みます。 → P.15 基礎知識編３．１０参照

緯度読込み

経度読込み

```
186   // read array: latlon for 89a
187   // read lat
188   dhnd = H5Dopen(fhnd ,"Latitude of Observation Point for 89A", H5P_DEFAULT);
189   ret = H5Dread(dhnd, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL, H5P_DEFAULT, lat89a);
190   if(ret < 0){
        printf("H5Dread error: Latitude of Observation Point for 89A¥n");
        exit(1);
193   }
194   ret = H5Dclose(dhnd);
195   // read lon
196   dhnd = H5Dopen(fhnd ,"Longitude of Observation Point for 89A", H5P_DEFAULT);
197   ret = H5Dread(dhnd, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL, H5P_DEFAULT, lon89a);
198   if(ret < 0){
199     printf("H5Dread error: Longitude of Observation Point for 89A¥n");
      exit(1);
201   }
202   ret = H5Dclose(dhnd);
203   // sample display
204   printf("latlon89a[scan=0][pixel=0]: (%9.4f,%9.4f)¥n", lat89a[0][0],
lon89a[0][0]);
```

◇物理量データ読込み　※左側の数字は行番号です

物理量データを読込みます。

物理量は格納型が 2byte 符号あり整数(-32768～32767)で、スケール値が設定されているので、
物理量に付属しているアトリビュートにアクセスして、スケール値を読み出します。
スケールを適用する際は、欠損値(-32768)と異常値(-32767)は除外します。

スケール読込み

読込み

スケール処理

このスケール処理は AMTK を使用する場合は
自動的に行われます。
→ P.16 基礎知識編３．１２参照

欠損値(-32768)と異常値(-32767)は除外して、
スケールを適用します。

```
226   // read array: geophysical data for 1 layer for 89a
227   dhnd = H5Dopen(fhnd, "Geophysical Data for 89A", H5P_DEFAULT);
228   // get scale
229   ahnd = H5Aopen(dhnd, "SCALE FACTOR", H5P_DEFAULT);
230   ret = H5Aread(ahnd, H5T_NATIVE_FLOAT, &sca);
231   ret = H5Aclose(ahnd);
232   // read
233   ret = H5Dread(dhnd, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL, H5P_DEFAULT,
geo1_89a);
      if(ret < 0){
        printf("H5Dread error: Geophysical Data for 89A¥n");
236     exit(1);
237   }
238   ret = H5Dclose(dhnd);
239   // change scale
240   for(j = 0; j < num; ++j){
        for(i = 0; i < AM2_DEF_SNUM_HI; ++i){
          if(geo1_89a[j][i] > -32767) geo1_89a[j][i] = geo1_89a[j][i] * sca;
243     }
244   }
```

◇L2 品質フラグデータ読込み　※左側の数字は行番号です

L2 品質フラグデータを読込みます。

```
266   // read array: pixel data quality for 1 layer for 89a
267   dhnd = H5Dopen(fhnd, "Pixel Data Quality for 89A", H5P_DEFAULT);
268   ret = H5Dread(dhnd, H5T_NATIVE_UCHAR, H5S_ALL, H5S_ALL, H5P_DEFAULT,
pdq1_89a);
269   if(ret < 0){
270     printf("H5Dread error: Pixel Data Quality for 89A¥n");
271     exit(1);
272   }
273   ret = H5Dclose(dhnd);
```

L2 品質フラグには、アルゴリズム開発者が設定した物理量算出に係る補足情報が格納されています。
0～15 は OK 状態、16～255 は NG 状態を現します。
NG 状態の場合、物理量には欠損値(-32768)または異常値(-32767)が格納されています。
L2 品質フラグの詳細については、「AMSR2 高次プロダクトフォーマット説明書」(*1)を参照ください。
(*1)http://suzaku.eorc.jaxa.jp/GCOM_W/data/data_w_format_j.html

◇終了処理　※左側の数字は行番号です

ファイルをクローズします。
HDF5 を終了します。

```
314   // close
315   ret = H5Fclose(fhnd);
316   ret = H5close();
```

◇ファイルのクローズ
ret = H5Fclose(fhnd);
fhnd:ファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇HDF5 の終了
ret = H5close();
ret: [戻り値]失敗の場合は負の値

７．４．２ コンパイル方法(build_readL2H_hdf5_c.sh 解説)

コンパイルに使用するスクリプト build_readL2H_hdf5_c.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～8 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 8 SZIP=/home/user1/util/szip_2.1
 9
```

11 行目に、使用するコンパイラを指定します。
インテルコンパイラ(icc)または PG コンパイラ(pgcc)または GNU コンパイラ(gcc)を指定します。

```
10 # compiler
11 cc=icc
12
13 # source filename
14 csrc="readL2H_hdf5.c amsr2time_.c "
15
16 # output filename
17 out=readL2H_hdf5_c
18
19 # library order
20 lib="-lhdf5_hl -lhdf5 -lsz -lz -lm"
21
22 # c compile
23 cmd="$cc -g $csrc -o $out -I$HDF5/include -I$SZIP/include -L$HDF5/lib
-L$SZIP/lib $lib"
24 echo $cmd
25 $cmd
26
27 # garbage
28 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL2H_hdf5_c.sh
icc -g readL2H_hdf5.c amsr2time_.c -o readL2H_hdf5_c
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lhdf5_hl -lhdf5 -lsz -lz -lm
~
```

７．４．３ プログラム実行結果のサンプル

サンプルプログラムでは固定配列を多数使用しているため、環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞

```
$ unlimit
```

＜sh/bash 環境の場合＞

※以下のコマンドを順番に、4 つとも実行してください。

```
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited
```

```
$ ./readL2H_hdf5_c GW1AM2_201303011809_125D_L2SGPRCHA0000000.h5
input file: GW1AM2_201303011809_125D_L2SGPRCHA0000000.h5
GeophysicalName: Precipitation
GranuleID: GW1AM2_201303011809_125D_L2SGPRCHA0000000
ObservationStartDateTime: 2013-03-01T18:09:10.122Z
EquatorCrossingDateTime: 2013-03-01T18:35:54.849Z
ObservationEndDateTime: 2013-03-01T18:58:26.342Z
NumberOfScans: 1972
limit of NumberOfScans = 2200
OverlapScans: 0
amsr2time: AMSR2_LEAP_DATA = /export/emc3/util/common/AMTK_AMSR2_DATA/leapsec.dat
amsr2time: year=1993 month= 7 tai93sec=   15638401.00
amsr2time: year=1994 month= 7 tai93sec=   47174402.00
amsr2time: year=1996 month= 1 tai93sec=   94608003.00
amsr2time: year=1997 month= 7 tai93sec=  141868804.00
amsr2time: year=1999 month= 1 tai93sec=  189302405.00
amsr2time: year=2006 month= 1 tai93sec=  410227206.00
amsr2time: year=2009 month= 1 tai93sec=  504921607.00
amsr2time: year=2012 month= 7 tai93sec=  615254408.00
amsr2time: number of leap second = 8
time[scan=0]: 2013/03/01 18:09:10
latlon89a[scan=0][pixel=0]: (  84.4188, -77.9502)
latlon89b[scan=0][pixel=0]: (  84.3305, -78.8925)
geo1_89a[scan=0][pixel=0]: -32767.0 [mm/h] (PDQ: 16)
geo1_89b[scan=0][pixel=0]: -32767.0 [mm/h] (PDQ: 16)
```

７．５ L3 輝度温度データ読込み

７．５．１ サンプルプログラム readL3B_hdf5.c 解説

サンプルプログラム readL3B_hdf5.c では、L3 輝度温度データファイルから、以下のメタデータと格納データを読込んで、内容をテキスト表示します。

メタデータ	格納データ
* GeophysicalName - GranuleID	* Brightness Temperature (H) - Brightness Temperature (V)

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の HDF5 関数は、最初に使用する際に使用説明を記載します。

HDF5 ライブラリでは、ファイルをオープンした後に、各データについてもオープンとクローズを繰り返します。各データの読込みは以下のような流れになります。

データをオープンする →必要な場合はスケールを取得する
→ データを読込む → データをクローズする

◇変数宣言　※左側の数字は行番号です

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <string.h>
 4 #include "hdf5.h"
 :
 9 int main(int argc, char *argv[]){
10   // interface variable
11   int i,j;         // loop variable
12   herr_t ret;      // return status
13   char *buf = NULL; // text buffer
14   char *fn  = NULL; // filename
15   hid_t fhnd;      // file handle
16   hid_t ahnd;      // attribute handle
17   hid_t atyp;      // attribute type
18   hid_t dhnd;      // dataset handle
19   hid_t shnd;      // dataspace handle
20   hsize_t sz1[3];   // array size 1
21   hsize_t sz2[3];   // array size 2
22   int x;           // grid size x
23   int y;           // grid size y
 :
31   // meta data
32   char *geo = NULL; // GeophysicalName
33   char *gid = NULL; // GranuleID
34
 :
38   // array data
39   float *tbH; // brightness temperature for horizontal
40   float *tbV; // brightness temperature for vertical
```

HDF5 用ヘッダーファイルをインクルードします。

HDF5 用インターフェイス
変数を宣言します。

物理量名
グラニュール ID

メタデータ用変数
を宣言します。

輝度温度

配列データ用変数
を宣言します。

この時点では、配列データ用のメモリ領域は確保されていません。
後程、配列サイズを調べてから、メモリ領域の確保を行います。

◇開始処理　※左側の数字は行番号です

HDF5 の初期化処理を行います。

◇HDF5 の初期化
ret = H5open();
ret: [戻り値]失敗の場合は負の値

```
52   // hdf5 initialize
53   ret = H5open();
54   if(ret < 0){
55     printf("h5open error: %d¥n" ,ret);
56     exit(1);
57   }
```

ファイルをオープンします。

◇HDF5 ファイルオープン
fhnd = H5Fopen(fn, label1, label2);
fn: オープンするファイル名を指定します
label1: アクセスモードを指定します H5F_ACC_RDONLY で読込専用になります
label2: H5P_DEFAULT を指定します
fhnd: [戻り値]開いたファイルのハンドル値、失敗の場合は負の値

```
59   // open
60   fhnd = H5Fopen(fn, H5F_ACC_RDONLY, H5P_DEFAULT);
61   if(fhnd < 0){
62     printf("H5Fopen error: %s¥n", fn);
63     exit(1);
64   }
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込むには、ファイルに付属しているアトリビュートを H5Aopen()でオープンします。
アトリビュートの型を調べてからデータを読込みます。
不要になったハンドルはクローズします。

◇アトリビュートのオープン
ahnd = H5Aopen(fhnd, nam, label);
fhnd: ファイルハンドル値を指定します
nam: アトリビュート名を指定します
label: H5P_DEFAULT を指定します
ahnd: [戻り値]アトリビュートのハンドル値
失敗の場合は負の値

◇アトリビュートタイプの取得
atyp = H5Aget_type(ahnd);
ahnd: アトリビュートハンドル値を指定します
atyp: [戻り値]アトリビュートタイプ値
失敗の場合は負の値

◇アトリビュート読込み
ret = H5Aread(ahnd, otyp, buf);
ahnd: アトリビュートハンドル値を指定します
otyp: 出力変数の型を指定します(読込んだデータは、出力変数の型へ自動変換されます)
buf: 出力変数を指定します
ret: [戻り値]失敗の場合は負の値

```
66   // read meta: GeophysicalName
67   ahnd = H5Aopen(fhnd, "GeophysicalName", H5P_DEFAULT);
68   atyp = H5Aget_type(ahnd);
69   ret = H5Aread(ahnd, atyp, &geo);
70   if(ret < 0){
71     printf("H5Aread error: GeophysicalName¥n");
72     exit(1);
73   }
74   ret = H5Aclose(ahnd);
75   printf("GeophysicalName: %s¥n", geo);
```

◇アトリビュートのクローズ
ret = H5Aclose(ahnd);
ahnd:アトリビュートハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇配列サイズ取得とメモリ領域確保　※左側の数字は行番号です

配列サイズを取得します。

◇データセットのオープン
dhnd = H5Dopen(fhnd, nam, label);
fhnd: ファイルハンドル値を指定します
nam: データセット名を指定します
label: H5P_DEFAULT を指定します
dhnd: [戻り値]データセットのハンドル値
失敗の場合は負の値

◇データスペースのオープン
shnd = H5Dget_space(dhnd)
dhnd: データセットハンドル値を指定します
shnd: [戻り値]データスペースのハンドル値
失敗の場合は負の値

◇データセットのサイズ取得
ret = H5Sget_simple_extent_dims(shnd,sz1,sz2)
shnd: データスペースハンドル値を指定します
sz1: [戻り値]データセットの各次元サイズ
sz2: [戻り値]データセットの各次元の最大サイズ
ret: [戻り値]失敗の場合は負の値

```
101   // get grid size
102   dhnd = H5Dopen(fhnd ,"Brightness Temperature (H)", H5P_DEFAULT);
103   shnd = H5Dget_space(dhnd);
104   ret = H5Sget_simple_extent_dims(shnd,sz1,sz2);
105   if(ret < 0){
106     printf("H5Sget_simple_extent_dims error: Brightness Temperature (H)¥n");
107     exit(1);
108   }
109   ret = H5Sclose(shnd);
110   ret = H5Dclose(dhnd);
111   x=sz1[1];
112   y=sz1[0];
113   printf("grid size x: %d¥n", x);
114   printf("grid size y: %d¥n", y);
115
```

◇データスペースのクローズ
ret = H5Sclose(shnd)
shnd: データスペースハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇データセットのクローズ
ret = H5Dclose(dhnd)
dhnd: データセットハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

メモリ領域を確保します。

```
116   // memory allocate
117   tbH=malloc(sizeof(float)*x*y);
118   if(tbH==NULL){
119     printf("memory allocate error: tbH¥n");
120     exit(1);
121   }
```

◇輝度温度データ読込み　※左側の数字は行番号です

輝度温度データを読込みます。

輝度温度は格納型が 2byte 符号なし整数(0～65535)で、スケール値が設定されているので、
輝度温度に付属しているアトリビュートにアクセスして、スケール値を読み出します。
スケールを適用する際は、欠損値(65535)と異常値(65534)は除外します。

◇データ読込み
ret = H5Dread(dhnd, otyp, label1, label2, label3, buf);
dhnd: データセットハンドル値を指定します
otyp: 出力変数の型を指定します(読込んだデータは、出力変数の型へ自動変換されます)
label1,label2: 部分配列を読込む場合に使用します
全て読込む場合は、どちらも H5S_ALL を指定します
label3: H5P_DEFAULT を指定します
buf: 出力変数を指定します
ret: [戻り値]失敗の場合は負の値

```
128   // read horizontal
129   dhnd = H5Dopen(fhnd, "Brightness Temperature (H)", H5P_DEFAULT);
130   // get scale
131   ahnd = H5Aopen(dhnd, "SCALE FACTOR", H5P_DEFAULT);
132   ret = H5Aread(ahnd, H5T_NATIVE_FLOAT, &sca);
133   ret = H5Aclose(ahnd);
134   // read
135   ret = H5Dread(dhnd, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL, H5P_DEFAULT, tbH);
136   if(ret < 0){
137     printf("H5Dread error: Brightness Temperature (H)¥n");
138     exit(1);
139   }
140   ret = H5Dclose(dhnd);
141   // change scale
142   for(j = 0; j < y; ++j){
143     for(i = 0; i < x; ++i){
144       if(tbH[x*j+i] < 65534) tbH[x*j+i] = tbH[x*j+i] * sca;
145     }
146   }
```

スケール読込み

読込み

スケール処理

このスケール処理は AMTK を使用する場合は
自動的に行われます。
→ P.16 基礎知識編３．１２参照

欠損値(65535)と異常値(65534)は除外して、
スケールを適用します。

◇終了処理　※左側の数字は行番号です

メモリを開放します。

```
252   // memory free
253   free(tbH);
254   free(tbV);
```

ファイルをクローズします。HDF5 を終了します。

◇ファイルのクローズ
ret = H5Fclose(fhnd);
fhnd:ファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇HDF5 の終了
ret = H5close();
ret: [戻り値]失敗の場合は負の値

```
256   // close
257   ret = H5Fclose(fhnd);
258   ret = H5close();
```

７．５．２ コンパイル方法(build_readL3B_hdf5_c.sh 解説)

コンパイルに使用するスクリプト build_readL3B_hdf5_c.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～8 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 8 SZIP=/home/user1/util/szip_2.1
 9
```

11 行目に、使用するコンパイラを指定します。
インテルコンパイラ(icc)または PG コンパイラ(pgcc)または GNU コンパイラ(gcc)を指定します。

```
10 # compiler
11 cc=icc
12
13 # source filename
14 csrc=readL3B_hdf5.c
15
16 # output filename
17 out=readL3B_hdf5_c
18
19 # library order
20 lib="-lhdf5_hl -lhdf5 -lsz -lz -lm"
21
22 # c compile
23 cmd="$cc -g $csrc -o $out -I$HDF5/include -I$SZIP/include -L$HDF5/lib
-L$SZIP/lib $lib"
24 echo $cmd
25 $cmd
26
27 # garbage
28 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL3B_hdf5_c.sh
icc -g readL3B_hdf5.c -o readL3B_hdf5_c
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lhdf5_hl -lhdf5 -lsz -lz -lm
~
```

７．５．３ プログラム実行結果のサンプル

環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞
$ unlimit

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、
4 つとも実行してください。
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited

```
$ ./readL3B_hdf5_c GW1AM2_20130200_01M_EQMA_L3SGT06LA1110110.h5
input file: GW1AM2_20130200_01M_EQMA_L3SGT06LA1110110.h5
GeophysicalName: Brightness Temperature (6GHz)
GranuleID: GW1AM2_20130200_01M_EQMA_L3SGT06LA1110110
grid size x: 1440
grid size y: 720

ASCII ART OF HORIZONTAL BRIGHTNESS TEMPERATURE (X/20GRID Y/40GRID)
+------------------------------------------------------------------------+
|                                                                        |
|4112333444434444444444444444444444444444444444444444444444443344322222244444|
|1111343111134444444444444444444444444444444444444444444444444434233343111|
|1245234455555555444444444444444434312343444144555444444444442112111111111|
|1545444444444444444444444445441111111111111111113554444444442221111111112|
|4122425544144445444444453411311111111111111111111455544445511111111111115|
|4444444442444555445555441111111111111111111111155555351111111111111124|
|4444444424441115551555311111111111111111121111111155151141111111111444|
|5555555555511111211113111211111111111111111111111111111141554111111111155|
|1155555551111111111151553111111111111111111111111111111145555541111111111|
|1115555531111111111111114211113121111111111111111111111155555555111111111|
|1115555215111111111111114555551111111111111111111111111145555521111111|
|1111555111111111111111135555555111111111111111111111111154552111111111|
|1111111111111111111111111111111111131111111111111111111155111111111111|
|1111111111111111111111111111111111111111111111111111111135111111111111|
|1111111111111111111111111111111111111111111111111111111111111111111111|
|3433234443333424343343444444443343321111111111111111111344444441111133|
|3333333333333333333333333333333343333333222223333333443344333333333333333|
+------------------------------------------------------------------------+
```

```
ASCII ART OF VERTICAL BRIGHTNESS TEMPERATURE (X/20GRID Y/40GRID)
+------------------------------------------------------------------------+
|                                                                        |
|43334444444445545544444444444444444444445444444444444444444433333244545|
|33334543333455455444454444444444444444444454544444444444444455443443443333|
|33553445555555555555455544545555444334545543455555444455444433333333333|
|35555555555555555545555555555433333333333333333345555555555433333333333333|
|43334355553555555555555545334333333333333333333355555555553333333333333335|
|55555555554555555555555553333333333333333333333355555453333333333333335|
|55555555355533355535554333333333333333333333333333553533433333333333555|
|65556665555333333333433343333333333333333333333333333535555333333333355|
|33555555633333333335355433333333333333333333333333333355555554333333333|
|33355555433333333333333433333433333333333333333333333355555556333333333|
|33355553353333333333333566665333333333333333333333333355555543333333333|
|33336553333333333333346666665333333333333333333333333355554333333333333|
|33333333333333333333333333333333343333333333333333333355333333333333333|
|33333333333333333333333333333333333333333333333333333345333333333333333|
|33333333333333333333333333333333333333333333333333333333333333333333333|
|44433444444444344444444444444444444433333333333333333334445555433333344|
|44444443344444444443333443444444444444444334444444444444444444444444444|
+------------------------------------------------------------------------+

[#]:missing
[ ]:out of observation
[1]: 50-100K
[2]:100-150K
[3]:150-200K
[4]:200-250K
[5]:250-300K
[6]:300-350K
[*]:other
```

７．６ L3 物理量データ読込み

７．６．１ サンプルプログラム readL3G_hdf5.c 解説

サンプルプログラム readL3G_hdf5.c では、L3 物理量データファイルから、以下のメタデータと格納データを読込んで、内容をテキスト表示します。

メタデータ	格納データ
* GeophysicalName - GranuleID	* Geophysical Data

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の HDF5 関数は、最初に使用する際に使用説明を記載します。

HDF5 ライブラリでは、ファイルをオープンした後に、各データについてもオープンとクローズを繰り返します。各データの読込みは以下のような流れになります。

データをオープンする →必要な場合はスケールを取得する
→ データを読込む → データをクローズする

◇変数宣言　※左側の数字は行番号です

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <string.h>
 4 #include "hdf5.h"
 :
 9 int main(int argc, char *argv[]){
10   // interface variable
11   int i,j;         // loop variable
12   herr_t ret;      // return status
13   char *buf = NULL; // text buffer
14   char *fn  = NULL; // filename
15   hid_t fhnd;      // file handle
16   hid_t ahnd;      // attribute handle
17   hid_t atyp;      // attribute type
18   hid_t dhnd;      // dataset handle
19   hid_t shnd;      // dataspace handle
20   hsize_t sz1[3];   // array size 1
21   hsize_t sz2[3];   // array size 2
22   int x;           // grid size x
23   int y;           // grid size y
 :
33   // meta data
34   char *geo = NULL; // GeophysicalName
35   char *gid = NULL; // GranuleID
 :
40   // array data
41   float *geo1; // geophysical data layer 1
42   float *geo2; // geophysical data layer 2
43   float *geotmp; // geophysical data temporary
```

HDF5 用ヘッダーファイルをインクルードします。

HDF5 用インターフェイス
変数を宣言します。

物理量名
グラニュール ID

メタデータ用変数
を宣言します。

輝度温度

配列データ用変数
を宣言します。

この時点では、配列データ用のメモリ領域は確保されていません。
後程、配列サイズを調べてから、メモリ領域の確保を行います。

◇開始処理　※左側の数字は行番号です

HDF5 の初期化処理を行います。

◇HDF5 の初期化
ret = H5open();
ret: [戻り値]失敗の場合は負の値

```
55   // hdf5 initialize
56   ret = H5open();
57   if(ret < 0){
58     printf("h5open error: %d¥n" ,ret);
59     exit(1);
60   }
```

ファイルをオープンします。

◇HDF5 ファイルオープン
fhnd = H5Fopen(fn, label1, label2);
fn: オープンするファイル名を指定します
label1: アクセスモードを指定します H5F_ACC_RDONLY で読込専用になります
label2: H5P_DEFAULT を指定します
fhnd: [戻り値]開いたファイルのハンドル値、失敗の場合は負の値

```
62   // open
63   fhnd = H5Fopen(fn, H5F_ACC_RDONLY, H5P_DEFAULT);
64   if(fhnd < 0){
65     printf("H5Fopen error: %s¥n", fn);
66     exit(1);
67   }
```

◇メタデータ読込み　※左側の数字は行番号です

メタデータを読込むには、ファイルに付属しているアトリビュートを H5Aopen()でオープンします。
アトリビュートの型を調べてからデータを読込みます。
不要になったハンドルはクローズします。

◇アトリビュートのオープン
ahnd = H5Aopen(fhnd, nam, label);
fhnd: ファイルハンドル値を指定します
nam: アトリビュート名を指定します
label: H5P_DEFAULT を指定します
ahnd: [戻り値]アトリビュートのハンドル値
失敗の場合は負の値

◇アトリビュートタイプの取得
atyp = H5Aget_type(ahnd);
ahnd: アトリビュートハンドル値を指定します
atyp: [戻り値]アトリビュートタイプ値
失敗の場合は負の値

◇アトリビュート読込み
ret = H5Aread(ahnd, otyp, buf);
ahnd: アトリビュートハンドル値を指定します
otyp: 出力変数の型を指定します(読込んだデータは、出力変数の型へ自動変換されます)
buf: 出力変数を指定します
ret: [戻り値]失敗の場合は負の値

```
69   // read meta: GeophysicalName
70   ahnd = H5Aopen(fhnd, "GeophysicalName", H5P_DEFAULT);
71   atyp = H5Aget_type(ahnd);
72   ret = H5Aread(ahnd, atyp, &geo);
73   if(ret < 0){
74     printf("H5Aread error: GeophysicalName¥n");
75     exit(1);
76   }
77   ret = H5Aclose(ahnd);
78   printf("GeophysicalName: %s¥n", geo);
```

◇アトリビュートのクローズ
ret = H5Aclose(ahnd);
ahnd:アトリビュートハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇配列サイズ取得とメモリ領域確保　※左側の数字は行番号です

配列サイズを取得します。

◇データセットのオープン
dhnd = H5Dopen(fhnd, nam, label);
fhnd: ファイルハンドル値を指定します
nam: データセット名を指定します
label: H5P_DEFAULT を指定します
dhnd: [戻り値]データセットのハンドル値
失敗の場合は負の値

◇データスペースのオープン
shnd = H5Dget_space(dhnd)
dhnd: データセットハンドル値を指定します
shnd: [戻り値]データスペースのハンドル値
失敗の場合は負の値

◇データセットのサイズ取得
ret = H5Sget_simple_extent_dims(shnd,sz1,sz2)
shnd: データスペースハンドル値を指定します
sz1: [戻り値]データセットの各次元サイズ
sz2: [戻り値]データセットの各次元の最大サイズ
ret: [戻り値]失敗の場合は負の値

```
105   // get grid size
106   dhnd = H5Dopen(fhnd ,"Geophysical Data", H5P_DEFAULT);
107   shnd = H5Dget_space(dhnd);
108   ret = H5Sget_simple_extent_dims(shnd,sz1,sz2);
109   if(ret < 0){
110     printf("H5Sget_simple_extent_dims error: Geophysical Data¥n");
111     exit(1);
112   }
113   ret = H5Sclose(shnd);
114   ret = H5Dclose(dhnd);
115   x=sz1[1];
116   y=sz1[0];
117   printf("grid size x: %d¥n", x);
118   printf("grid size y: %d¥n", y);
```

◇データスペースのクローズ
ret = H5Sclose(shnd)
shnd: データスペースハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇データセットのクローズ
ret = H5Dclose(dhnd)
dhnd: データセットハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

メモリ領域を確保します。

```
120   // memory allocate layer 1
121   geo1=malloc(sizeof(float)*x*y);
122   if(geo1==NULL){
123     printf("memory allocate error: geo1¥n");
124     exit(1);
125   }
```

◇物理量データ読込み　※左側の数字は行番号です

物理量データを読込みます。
積雪深(SND)・海面水温(SST)は物理量が 2 層あるので、この 2 つとそれ以外で処理を分けています。
積雪深の 2 層目は積雪水量[cm]です。海面水温の 2 層目は 10GHz による SST[℃]です。

物理量は格納型が 2byte 符号あり整数(-32768～32767)で、スケール値が設定されているので、
物理量に付属しているアトリビュートにアクセスして、スケール値を読み出します。
スケールを適用する際は、欠損値(-32768)と異常値(-32767)は除外します。

◇データ読込み
ret = H5Dread(dhnd, otyp, label1, label2, label3, buf);
dhnd: データセットハンドル値を指定します
otyp: 出力変数の型を指定します(読込んだデータは、出力変数の型へ自動変換されます)
label1,label2: 部分配列を読込む場合に使用します
　　全て読込む場合は、どちらも H5S_ALL を指定します
label3: H5P_DEFAULT を指定します
buf: 出力変数を指定します
ret: [戻り値]失敗の場合は負の値

```
141   // read layer 1
142   if(strncmp(gid+29,"SND",3)!=0 && strncmp(gid+29,"SST",3)!=0 ){
143     dhnd = H5Dopen(fhnd, "Geophysical Data", H5P_DEFAULT);
144     // get scale
145     ahnd = H5Aopen(dhnd, "SCALE FACTOR", H5P_DEFAULT);
146     ret = H5Aread(ahnd, H5T_NATIVE_FLOAT, &sca);
147     ret = H5Aclose(ahnd);
148     // read
149     ret = H5Dread(dhnd, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL, H5P_DEFAULT, geo1);
150     if(ret < 0){
          printf("H5Dread error: Geophysical Data¥n");
152       exit(1);
153     }
154     ret = H5Dclose(dhnd);
155     // change scale
156     for(j = 0; j < y; ++j){
157       for(i = 0; i < x; ++i){
            if(geo1[x*j+i] > -32767) geo1[x*j+i] = geo1[x*j+i] * sca;
159       }
160     }
161   }
```

スケール読込み

読込み

スケール処理

このスケール処理は AMTK を使用する場合は
自動的に行われます。
→ P.16 基礎知識編３．１２参照

欠損値(-32768)と異常値(-32767)は除外して、
スケールを適用します。

2 層ある場合は、一時変数で全体を読込んだ後に、層ごとに分割しています。

```
163   // read layer 1 & 2
164   if(strncmp(gid+29,"SND",3)==0 || strncmp(gid+29,"SST",3)==0 ){
165     dhnd = H5Dopen(fhnd, "Geophysical Data", H5P_DEFAULT);
166     // get scale
167     ahnd = H5Aopen(dhnd, "SCALE FACTOR", H5P_DEFAULT);
168     ret = H5Aread(ahnd, H5T_NATIVE_FLOAT, &sca);
169     ret = H5Aclose(ahnd);
170     // read
171     ret = H5Dread(dhnd, H5T_NATIVE_FLOAT, H5S_ALL, H5S_ALL, H5P_DEFAULT,
geotmp);
172     if(ret < 0){
173       printf("H5Dread error: Geophysical Data¥n");
174       exit(1);
175     }
176     ret = H5Dclose(dhnd);
177     // separate
178     for(j = 0; j < y; ++j){
179       for(i = 0; i < x; ++i){
180         geo1[x*j+i] = geotmp[x*j*2+i*2+0];
181         geo2[x*j+i] = geotmp[x*j*2+i*2+1];
182       }
183     }
184     // change scale
185     for(j = 0; j < y; ++j){
186       for(i = 0; i < x; ++i){
187         if(geo1[x*j+i] > -32767) geo1[x*j+i] = geo1[x*j+i] * sca;
188         if(geo2[x*j+i] > -32767) geo2[x*j+i] = geo2[x*j+i] * sca;
189       }
190     }
191   }
```

◇終了処理　※左側の数字は行番号です

メモリを開放します。

```
373   // memory free
374   free(geo1);
375   if(strncmp(gid+29,"SND",3)==0 || strncmp(gid+29,"SST",3)==0 ){
376     free(geo2);
377   }
```

ファイルをクローズします。HDF5 を終了します。

◇ファイルのクローズ
ret = H5Fclose(fhnd);
fhnd:ファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇HDF5 の終了
ret = H5close();
ret: [戻り値]失敗の場合は負の値

```
379   // close
380   ret = H5Fclose(fhnd);
381   ret = H5close();
```

７．６．２ コンパイル方法(build_readL3G_hdf5_c.sh 解説)

コンパイルに使用するスクリプト build_readL3G_hdf5_c.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～8 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 8 SZIP=/home/user1/util/szip_2.1
 9
```

11 行目に、使用するコンパイラを指定します。
インテルコンパイラ(icc)または PG コンパイラ(pgcc)または GNU コンパイラ(gcc)を指定します。

```
10 # compiler
11 cc=icc
12
13 # source filename
14 csrc=readL3G_hdf5.c
15
16 # output filename
17 out=readL3G_hdf5_c
18
19 # library order
20 lib="-lhdf5_hl -lhdf5 -lsz -lz -lm"
21
22 # c compile
23 cmd="$cc -g $csrc -o $out -I$HDF5/include -I$SZIP/include -L$HDF5/lib
-L$SZIP/lib $lib"
24 echo $cmd
25 $cmd
26
27 # garbage
28 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL3G_hdf5_c.sh
icc -g readL3G_hdf5.c -o readL3G_hdf5_c
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lhdf5_hl -lhdf5 -lsz -lz -lm
~
```

７．６．３ プログラム実行結果のサンプル

環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞

```
$ unlimit
```

＜sh/bash 環境の場合＞

※以下のコマンドを順番に、4 つとも実行してください。

```
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited
```

```
$ ./readL3G_hdf5_c GW1AM2_20130200_01M_EQMA_L3SGCLWLA0000000.h5
input file: GW1AM2_20130200_01M_EQMA_L3SGCLWLA0000000.h5
GeophysicalName: Cloud Liquid Water
GranuleID: GW1AM2_20130200_01M_EQMA_L3SGCLWLA0000000
grid size x: 1440
grid size y: 720

ASCII ART OF GEOPHYSICAL DATA LAYER #1 (X/20GRID Y/40GRID)
+------------------------------------------------------------------------+
|                                                                        |
|#21#####################################################################|
|2212###1113#############################################################222|
|1#################################11######3################011#42223322|
|2##############################00101112413333233##############343413131#|
|#20#######2###############00#12234523442132111 20#########3355421131121#|
|########################4#3222223232100110111110#######14*41111201111##|
|########0###000###0####1111111121012011 0*131121100##1#11#01211110100###|
|##########1011111111#211#11111210011121333223224321000#1###1111000000##|
|11#######00010122125#3###1#122323211000100201100010111 22######222321111|
|000######122131112311112#01112#113122123231101000000001 10########1001110|
|100####32#12111141111000######223232242121244133111001 1010######11111100|
|1000###121123221125110 0########131112232110121313122011010#####231111001|
|11131212231110111111211111110012102#123122122331121311 1021##011124223242|
|*33222511232222222223223331421022422322211024222222122113##0111421221132|
|10112211222111122211111111122231223223233124332121232322122331223222211 1|
|#####################################3111221222211121112 21########21221##|
|#####################################################################|
+------------------------------------------------------------------------+

[#]:missing
[ ]:out of observation
[0]:  0.000 -  0.030 Kg/m2
[1]:  0.030 -  0.060 Kg/m2
[2]:  0.060 -  0.090 Kg/m2
[3]:  0.090 -  0.120 Kg/m2
[4]:  0.120 -  0.150 Kg/m2
[5]:  0.150 -  0.180 Kg/m2
[*]:other
```

８．HDF5 編(FORTRAN90)

赤文字の解説はサンプルプログラムについて説明しています。

青文字の解説は関数リファレンスまたは衛星基礎知識について説明しています。

８．１ L1B データ読込み

８．１．１ サンプルプログラム readL1B_hdf5.f 解説

サンプルプログラム readL1B_hdf5.f では、L1B データファイルから、以下の格納データを読込んで、89G A ホーン緯度経度から低周波緯度経度を算出し、内容をテキスト表示します。
TAI93 形式時刻の変換と低周波緯度経度の算出には、サブルーチンを用意しています。

メタデータ	格納データ
※HDF5 の FORTRAN90 ライブラリでは可変長文字列メタデータを読込めません。 ＊ NumberOfScans は配列サイズから調べます。 以下の 3 つは固定値を使用します。 ＊ OverlapScans ＊ CoRegistrationParameterA1 ＊ CoRegistrationParameterA2	＊ Scan Time ＊ Latitude of Observation Point for 89A - Latitude of Observation Point for 89B - Longitude of Observation Point for 89A - Longitude of Observation Point for 89B ＊ Brightness Temperature (6.9GHz,H) - Brightness Temperature (6.9GHz,V) - Brightness Temperature (7.3GHz,H) - Brightness Temperature (7.3GHz,V) - Brightness Temperature (10.7GHz,H) - Brightness Temperature (10.7GHz,V) - Brightness Temperature (18.7GHz,H) - Brightness Temperature (18.7GHz,V) - Brightness Temperature (23.8GHz,H) - Brightness Temperature (23.8GHz,V) - Brightness Temperature (36.5GHz,H) - Brightness Temperature (36.5GHz,V) - Brightness Temperature (89.0GHz-A,H) - Brightness Temperature (89.0GHz-A,V) - Brightness Temperature (89.0GHz-B,H) - Brightness Temperature (89.0GHz-B,V) ＊ Pixel Data Quality 6 to 36 - Pixel Data Quality 89 ＊ Land_Ocean Flag 6 to 36 - Land_Ocean Flag 89 ＊ Earth Incidence - Earth Azimuth
89G A ホーン緯度経度から算出するデータ	
＊ 緯度経度(低周波平均) - 緯度経度(6G) - 緯度経度(7G) - 緯度経度(10G) - 緯度経度(18G) - 緯度経度(23G) - 緯度経度(36G) → P.15 基礎知識編３．１０参照	

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の HDF5 関数は、最初に使用する際に使用説明を記載します。

HDF5 ライブラリでは、ファイルをオープンした後に、各データについてもオープンとクローズを繰り返します。各データの読込みは以下のような流れになります。

データをオープンする →必要な場合はスケールを取得する
→ データを読込む → データをクローズする

◇変数宣言　※左側の数字は行番号です

```
 1      program main
 2      use hdf5
 3      implicit none
 4 C include
 5      include 'amsr2time_f.h'
 :
 8 C fixed value
 9      integer(4),parameter::LMT=2200 ! limit of NumberOfScans
10      integer(4),parameter::AM2_DEF_SNUM_HI=486 ! high resolution pixel width
11      integer(4),parameter::AM2_DEF_SNUM_LO=243 ! low  resolution pixel width
12 C interface variable
13      integer(4) i,j          ! loop variable
14      integer(4) ret          ! return status
15      character(len=512) buf  ! text buffer
16      character(len=512) fn   ! filename
17      integer(HID_T) fhnd     ! file handle
18      integer(HID_T) ahnd     ! attribute handle
19      integer(HID_T) atyp     ! attribute type
20      integer(HID_T) dhnd     ! dataset handle
21      integer(HID_T) dtyp     ! dataset type
22      integer(HID_T) shnd     ! dataspace handle
23      integer(HSIZE_T) sz1(3) ! array size 1
24      integer(HSIZE_T) sz2(3) ! array size 2
 :
35      integer(4) num          ! NumberOfScans
36      integer(4) ovr          ! OverlapScans
37      real(8) prm1(7)         ! CoRegistrationParameterA1
38      real(8) prm2(7)         ! CoRegistrationParameterA2
39      parameter(ovr=20)
 :

45      type(AM2_COMMON_SCANTIME) st(LMT) ! scantime
46      real(4) lat89a(AM2_DEF_SNUM_HI,LMT) ! lat for 89a
47      real(4) lat89b(AM2_DEF_SNUM_HI,LMT) ! lat for 89b
 :
55      real(4) lon89a(AM2_DEF_SNUM_HI,LMT) ! lon for 89a
56      real(4) lon89b(AM2_DEF_SNUM_HI,LMT) ! lon for 89b
 :
64      real(4) tb06h(AM2_DEF_SNUM_LO,LMT)  ! tb for 06h
65      real(4) tb06v(AM2_DEF_SNUM_LO,LMT)  ! tb for 06v
 :
80      integer(1) pdq06h(AM2_DEF_SNUM_LO,LMT)  ! pixel data quality for 06h
81      integer(1) pdq06v(AM2_DEF_SNUM_LO,LMT)  ! pixel data quality for 06v
 :
89      integer(1) lof06(AM2_DEF_SNUM_LO,LMT)   ! land ocean flag for 06
90      integer(1) lof07(AM2_DEF_SNUM_LO,LMT)   ! land ocean flag for 07
 :
98      real(4) ear_in(AM2_DEF_SNUM_LO,LMT) ! earth incidence
99      real(4) ear_az(AM2_DEF_SNUM_LO,LMT) ! earth azimuth
```

HDF5 用 use 文を宣言します。

時刻変換用サブルーチンヘッダーを
インクルードします。

標準プロダクトのスキャン数上限は
2200 で充分ですが、
準リアルプロダクトを使用する場合は、
2 周回程度繋がる場合があるので、
LMT=9000 としてください。
(標準の 4 パス分程度)

HDF5 用インターフェイス
変数を宣言します。

スキャン数
オーバーラップスキャン数
相対レジストレーション係数 A1
相対レジストレーション係数 A2

メタデータ用変数を宣言します。
※HDF5 の FORTRAN90 ライブラリでは
可変長文字列のメタデータを読込めな
いので、後で固定値を設定します。

時刻データには AM2_COMMON_SCANTIME 構造体を使います。
配列の次元数はデータによって異なります。
スキャン数はパスによって若干上下するので、上限を決めて宣言しておきます。
ここでは上限として LMT=2200 を設定しています。
AM2_DEF_SNUM_HI は高解像度ピクセル数(486)です。
AM2_DEF_SNUM_LO は低解像度ピクセル数(243)です。

時刻
緯度
経度
輝度温度
品質フラグ
陸海フラグ
観測入射角
観測方位角

配列データ用変数
を宣言します。

◇開始処理　※左側の数字は行番号です

HDF5 の初期化処理を行います。

◇HDF5 の初期化
call H5open_f(ret)
ret: [戻り値]失敗の場合は負の値

```
108 C hdf5 initialize
109       call H5open_f(ret)
110       if(ret.lt.0)then
111         write(*,'(a,i12)')'h5open_f error: ',ret
112         call exit(1)
113       endif
```

ファイルをオープンします。

◇HDF5 ファイルオープン
call H5Fopen_f(fn,label1,fhnd,ret,label2)
fn: オープンするファイル名を指定します
label1: アクセスモードを指定します H5F_ACC_RDONLY_F で読込専用になります
fhnd: [戻り値]開いたファイルのハンドル値
ret: [戻り値]失敗の場合は負の値
label2: H5P_DEFAULT_F を指定します

```
114 C open
115       call H5Fopen_f(fn,H5F_ACC_RDONLY_F,fhnd,ret,H5P_DEFAULT_F)
116       if(ret.lt.0)then
117         write(*,'(a,a)')'H5Fopen error: ',fn(1:len_trim(fn))
118         call exit(1)
119       endif
```

◇NumberOfScans と相対レジストレーション係数の設定　※左側の数字は行番号です

HDF5 の FORTRAN90 ライブラリでは可変長文字列のメタデータを読み込めないので、
NumberOfScans を配列サイズから調べます。ここでは"Scan Time"のサイズを調べています。
配列サイズを取得するには、データセットをオープンして、データスペースをオープンしてから、
データセットのサイズ取得関数を使用します。不要になったハンドルはクローズします。

◇データセットのオープン
call H5Dopen_f(fhnd,nam,dhnd,ret)
fhnd: ファイルハンドル値を指定します
nam: データセット名を指定します
dhnd: [戻り値]データセットのハンドル値
ret: [戻り値]失敗の場合は負の値

◇データスペースのオープン
call H5Dget_space_f(dhnd,shnd,ret)
dhnd: データセットハンドル値を指定します
shnd: [戻り値]データスペースのハンドル値
ret: [戻り値]失敗の場合は負の値

◇データセットのサイズ取得
call H5Sget_simple_extent_dims_f(shnd,sz1,sz2,ret)
shnd: データスペースハンドル値を指定します
sz1: [戻り値]データセットの各次元サイズ
sz2: [戻り値]データセットの各次元の最大サイズ
ret: [戻り値]失敗の場合は負の値

```
120 C HDF5 FORTRAN LIBRARY CAN'T RETRIEVE VARIABLE STRING !
121 C calculate NumberOfScans from array size and OverlapScans
122       call H5Dopen_f(fhnd,'Scan Time',dhnd,ret)
123       call H5Dget_space_f(dhnd,shnd,ret)
124       call H5Sget_simple_extent_dims_f(shnd,sz1,sz2,ret)
125       if(ret.lt.0)then
126         write(*,'(a)')'H5Sget_simple_extent_dims error: Scan Time'
127         call exit(1)
128       endif
129       call H5Sclose_f(shnd,ret)
130       call H5Dclose_f(dhnd,ret)
131       num=sz1(1)-ovr*2
132       write(*,'(a,i12)')'NumberOfScans(RETRIEVE BY ARRAY SIZE): ',num
133       write(*,'(a,i12)')'OverlapScans(FIXED VALUE): ',ovr
```

NumberOfScans は、
「配列のスキャン数－(OverlapScans×２)」となります。

◇データスペースのクローズ
call H5Sclose_f(shnd,ret)
shnd: データスペースハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇データセットのクローズ
call H5Dclose_f(dhnd,ret)
dhnd: データセットハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

相対レジストレーション係数を設定します。
相対レジストレーション係数は、低周波緯度経度を算出する際に必要になります。

```
141 C HDF5 FORTRAN LIBRARY CAN'T RETRIEVE VARIABLE STRING !
142 C use fixed value for CoRegistrationParameter
143       prm1(1)= 1.25000D0
144       prm1(2)= 1.00000D0
145       prm1(3)= 1.25000D0
146       prm1(4)= 1.25000D0
147       prm1(5)= 1.25000D0
148       prm1(6)= 1.00000D0
149       prm1(7)=(prm1(1)+prm1(2)+prm1(3)+prm1(4)+prm1(5)+prm1(6))/6.0D0
:
154       prm2(1)= 0.00000D0
155       prm2(2)=-0.10000D0
156       prm2(3)=-0.25000D0
157       prm2(4)= 0.00000D0
158       prm2(5)=-0.25000D0
159       prm2(6)= 0.00000D0
160       prm2(7)=(prm2(1)+prm2(2)+prm2(3)+prm2(4)+prm2(5)+prm2(6))/6.0D0
```

P.15 基礎知識編３．１０参照

◇時刻データ読込み　※左側の数字は行番号です

データを読込むためには、データセットを H5Dopen()でオープンして、データタイプを取得して、
読込み用変数の各次元サイズを配列変数に設定しておきます。
L1 データには前後にオーバーラップが含まれるので重複部分を取り除きます。
時刻データの格納形式は TAI93 形式なので年/月/日/時/分/秒に変換します。

P.11 基礎知識編３． ５参照

◇データ読込み
call H5Dread_f(dhnd,dtyp,buf,sz1,ret,label1,label2)
dhnd: データセットハンドル値を指定します
dtyp: データタイプ値を指定します
buf: 読込み用の変数を指定します
sz1: buf の各次元サイズを設定しておきます
ret: [戻り値]失敗の場合は負の値
label1,label2: 部分配列を読込む場合に使用します
全て読込む場合は、どちらも H5S_ALL_F を指定します

```
165 C read array: scantime
166       ! read
167       call H5Dopen_f(fhnd,'Scan Time',dhnd,ret)
168       sz1(1)=LMT
169       call H5Dread_f(dhnd
          +,H5T_NATIVE_DOUBLE,r8d1,sz1,ret,H5S_ALL_F,H5S_ALL_F)
          if(ret.lt.0)then
172         write(*,'(a,a)')'H5Dread error: ','Scan Time'
173         call exit(1)
174       endif
175       call H5Dclose_f(dhnd,ret)
176       ! cutoff overlap
177       do j=1,num
178         r8d1(j)=r8d1(j+ovr)
          enddo
180       do j=num+1,LMT
181         r8d1(j)=0
182       enddo
183       ! convert
          call amsr2time(num,r8d1,st)
          ! sample display
186       write(*,'(a,i4.4,"/",i2.2,"/",i2.2," ",i2.2,":",i2.2,":",i2.2)')
187      +'time(scan=1): '
188      +,st(1)%year
189      +,st(1)%month
190      +,st(1)%day
191      +,st(1)%hour
192      +,st(1)%minute
193      +,st(1)%second
```

読込み

重複除去

時刻変換

AMSR2 プロダクトの時刻格納形式は、TAI93 形式と呼ばれる、
うるう秒を含めた 1993/01/01 からの通算秒です。
このままでは日時情報として扱い難いので、
このサンプルプログラムでは年/月/日/時/分/秒に変換する
サブルーチンを用意しています。この時刻変換は AMTK を使用
する場合は自動的に行われます。
→ P.11 基礎知識編３． ６参照

◆時刻形式の変換
call amsr2time(num,in,out)
num: スキャン数を指定します
in: TAI93 形式の時刻データを指定します
out: [戻り値]AM2_COMMON_SCANTIME 構造体で年/月/日/時/分/秒データが返されます

◇緯度経度データ読込み　※左側の数字は行番号です

89G 緯度経度は格納データを読込みます。
二次元配列を読込む場合には、サイズ指定配列の 1 番目と 2 番目に値を設定します。
読込んだ後は、オーバーラップを取り除きます。

低周波緯度経度は、89G A ホーン緯度経度と相対レジストレーション係数から算出します。

```
194 C read array: latlon for 89a
195       ! read lat
196       call H5Dopen_f(fhnd
197      +,'Latitude of Observation Point for 89A',dhnd,ret)
198       sz1(1)=LMT
199       sz1(2)=AM2_DEF_SNUM_HI
200       call H5Dread_f(dhnd
201      +,H5T_NATIVE_REAL,lat89a,sz1,ret,H5S_ALL_F,H5S_ALL_F)
202       if(ret.lt.0)then
203         write(*,'(a,a)')'H5Dread error: '
204      +  ,'Latitude of Observation Point for 89A'
205         call exit(1)
206       endif
207       call H5Dclose_f(dhnd,ret)
208       ! read lon
209       call H5Dopen_f(fhnd
210      +,'Longitude of Observation Point for 89A',dhnd,ret)
211       sz1(1)=LMT
212       sz1(2)=AM2_DEF_SNUM_HI
213       call H5Dread_f(dhnd
214      +,H5T_NATIVE_REAL,lon89a,sz1,ret,H5S_ALL_F,H5S_ALL_F)
215       if(ret.lt.0)then
216         write(*,'(a,a)')'H5Dread err
217      +  ,'Longitude of Observation P
218         call exit(1)
219       endif
220       call H5Dclose_f(dhnd,ret)
221       ! cutoff overlap
222       do j=1,num
223         lat89a(:,j)=lat89a(:,j+ovr)
224         lon89a(:,j)=lon89a(:,j+ovr)
225       enddo
226       do j=num+1,LMT
227         lat89a(:,j)=0
228         lon89a(:,j)=0
229       enddo
230       ! sample display
231       write(*,'(a,"(",f9.4,",",f9.4,")")')'latlon89a(pixel=1,scan=1): '
232      +,lat89a(1,1),lon89a(1,1)
:
272 C read array: latlon for low mean
273       call amsr2latlon(num,prm1(7),prm2(7),lat89a,lon89a,latlm,lonlm)
274       write(*,'(a,"(",f9.4,",",f9.4,")")')'latlonlm(pixel=1,scan=1): '
275      +,latlm(1,1),lonlm(1,1)
```

緯度読込み

経度読込み

重複除去

低周波計算

AMSR2 センサには 6G/7G/10G/18G/23G/36G/89G の 7 つの観測周波数がありますが、それらの観測点緯度経度は正確には異なります。
AMSR2 プロダクトに格納されているのは 89G 緯度経度だけです。低周波の正確な緯度経度は 89G A ホーン緯度経度と相対レジストレーション係数から算出できます。
このサンプルプログラムでは低周波緯度経度を算出するためのサブルーチンを用意しています。AMTK を使用する場合は、低周波緯度経度は自動的に算出されます。
→ P.15 基礎知識編３． １０参照

◆低周波緯度経度算出
call amsr2latlon(num,prm1,prm2,lat89a,lon89a,latlow,lonlow)
num: スキャン数を指定します
prm1: 相対レジストレーション係数 A1 を指定します(6G/7G/10G/18G/23G/36G/平均)
prm2: 相対レジストレーション係数 A2 を指定します(6G/7G/10G/18G/23G/36G/平均)
lat89a: 89G A ホーン緯度データを指定します
lon89a: 89G A ホーン経度データを指定します
latlow: [戻り値]指定した低周波の緯度データが返されます
lonlow: [戻り値]指定した低周波の経度データが返されます

◇輝度温度データ読込み　※左側の数字は行番号です

輝度温度は格納型が 2byte 符号なし整数(0～65535)で、スケール値が設定されているので、
輝度温度に付属しているアトリビュートにアクセスして、スケール値を読み出します。
スケールを適用する際は、欠損値(65535)と異常値(65534)は除外します。

◇アトリビュートのオープン
call H5Aopen_f(dhnd,nam,ahnd,ret)
dhnd: データセットハンドル値を指定します
nam: アトリビュート名を指定します
ahnd: [戻り値]アトリビュートのハンドル値
ret: [戻り値]失敗の場合は負の値

◇アトリビュート読込み
call H5Aread_f(ahnd,atyp,buf,sz1,ret)
ahnd: アトリビュートハンドル値を指定します
otyp: 出力変数の型を指定します
buf: 出力変数を指定します
sz1: buf の各次元サイズを設定しておきます(スカラ読込みでは無視されます)
ret: [戻り値]失敗の場合は負の値

```
300 C read array: tb for 06h
301       ! read
302       call H5Dopen_f(fhnd
30       +,'Brightness Temperature (6.9GHz,H)',dhnd,ret)
30        call H5Aopen_f(dhnd,'SCALE FACTOR',ahnd,ret)     ! g
305       call H5Aread_f(ahnd,H5T_NATIVE_REAL,sca,sz1,ret) !
306       call H5Aclose_f(ahnd,ret)                        ! get scale
307       sz1(1)=LMT
308       sz1(2)=AM2_DEF_SNUM_LO
309       call H5Dread_f(dhnd
         +,H5T_NATIVE_REAL,tb06h,sz1,ret,H5S_ALL_F,H5S_ALL_F)
          if(ret.lt.0)then
312         write(*,'(a,a)')'H5Dread error: '
313       +  ,'Brightness Temperature (6.9GHz,H)'
31          call exit(1)
315       endif
316       call H5Dclose_f(dhnd,ret)
317       ! cutoff overlap & convert to unsignd & change scale
318       do j=1,num
319         do i=1,AM2_DEF_SNUM_LO
320           tb06h(i,j)=tb06h(i,j+ovr)
321           if(tb06h(i,j).lt.65534)tb06h(i,j)=tb06h(i,j)*sca
322         enddo
          enddo
          do j=num+1,LMT
32          tb06h(:,j)=0
326       enddo
327       ! sample display
 328       write(*,'(a,f9.2)')'tb06h(pixel=1,scan=1): ',tb06h(1,1)
```

スケール読込み

読込み

重複除去

欠損値(65535)と異常値(65534)は除外して、
スケールを適用します。

◇アトリビュートのクローズ
call H5Aclose_f(ahnd,ret)
ahnd:アトリビュートハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

このスケール処理は AMTK を使用する場合は
自動的に行われます。
→ P.16 基礎知識編３． １２参照

◇L1 品質フラグデータ読込み　※左側の数字は行番号です

L1 低周波品質フラグは 2 つの 1byte 整数型で 16 ビットの内 12 ビットが各低周波に対応します。
L1 高周波品質フラグは 1 つの 1byte 整数型で 8 ビットの内 4 ビットが各高周波に対応します。

読込み

```
764 C read array: pixel data quality for low
765       ! read
76        call H5Dopen_f(fhnd
76       +,'Pixel Data Quality 6 to 36',dhnd,ret)
768       sz1(1)=LMT
769       sz1(2)=AM2_DEF_SNUM_LO
          call H5Dread_f(dhnd
         +,H5T_NATIVE_INTEGER,i4d2hi,sz1,ret,H5S_ALL_F,H5S_ALL_F)
772       if(ret.lt.0)then
773         write(*,'(a,a)')'H5Dread error: '
774        +  ,'Pixel Data Quality 6 to 36'
77          call exit(1)
77        endif
777       call H5Dclose_f(dhnd,ret)
```

L1 品質フラグデータは、各周波数（6GHz、7GHz）、偏波に対する RFI 情報がまとまって取得されます。
まとまったままでは扱い難いので、各周波数毎の配列に分割します。
格納値は 2 ビットのマスク値なので、各周波数毎に integer(1)型の 2 次元配列を用意します。
L1 品質フラグデータには RFI(Radio Frequency Interference；電波干渉)情報が格納されています。
電波干渉の発生がない場合は 00、発生の可能性がある場合は 10、発生がある場合は 11 となります。

ビット毎の情報を各周波数に分解

重複除去も行います

```
778       ! cutoff overlap & separate
779       do j=1,num
780         do i=1,AM2_DEF_SNUM_LO
781          pdq06v(i,j)=0
78           if(btest(i4d2hi((i-1)*2+1,j+ovr),0))pdq06v(i,j)=1
             if(btest(i4d2hi((i-1)*2+1,j+ovr),1))pdq06v(i,j)=10
             if(btest(i4d2hi((i-1)*2+1,j+ovr),0)
            +    .and.btest(i4d2hi((i-1)*2+1,j+ovr),1))pdq06v(i,j)=11
             pdq06h(i,j)=0
787          if(btest(i4d2hi((i-1)*2+1,j+ovr),2))pdq06h(i,j)=1
             if(btest(i4d2hi((i-1)*2+1,j+ovr),3))pdq06h(i,j)=10
             if(btest(i4d2hi((i-1)*2+1,j+ovr),2)
            +    .and.btest(i4d2hi((i-1)*2+1,j+ovr),3))pdq06h(i,j)=11
791          pdq07v(i,j)=0
792          if(btest(i4d2hi((i-1)*2+1,j+ovr),4))pdq07v(i,j)=1
793          if(btest(i4d2hi((i-1)*2+1,j+ovr),5))pdq07v(i,j)=10
794          if(btest(i4d2hi((i-1)*2+1,j+ovr),4)
795         +    .and.btest(i4d2hi((i-1)*2+1,j+ovr),5))pdq07v(i,j)=11
796          pdq07h(i,j)=0
797          if(btest(i4d2hi((i-1)*2+1,j+ovr),6))pdq07h(i,j)=1
798          if(btest(i4d2hi((i-1)*2+1,j+ovr),7))pdq07h(i,j)=10
799          if(btest(i4d2hi((i-1)*2+1,j+ovr),6)
800         +    .and.btest(i4d2hi((i-1)*2+1,j+ovr),7))pdq07h(i,j)=11
801         enddo
80        enddo
80        do j=num+1,LMT
804         pdq06h(:,j)=0
805         pdq06v(:,j)=0
806         pdq07h(:,j)=0
807         pdq07v(:,j)=0
808       enddo
```

◇L1B 陸海フラグデータ読込み　※左側の数字は行番号です

L1B 陸海は integer(1)型の 2 次元配列((スキャン数*チャンネル数)×ピクセル数)です。
低周波データは周波数が 6 チャンネル(6G/7G/10G/18G/23G/36G)あります。
高周波データは周波数が 2 チャンネル(89G A ホーン/89G B ホーン)あります。

P. 14 基礎知識編３． ９参照

```
848 C read array: land ocean flag for low
849       ! read
850       call H5Dopen_f(fhnd
851      +,'Land_Ocean Flag 6 to 36',dhnd,ret)
852       sz1(1)=LMT*6
853       sz1(2)=AM2_DEF_SNUM_LO
          call H5Dread_f(dhnd
         +,H5T_NATIVE_INTEGER,loflo,sz1,ret,H5S_ALL_F,H5S_ALL_F)
856       if(ret.lt.0)then
857         write(*,'(a,a)')'H5Dread error: '
858        +  ,'Land_Ocean Flag 6 to 36'
859         call exit(1)
860       endif
861       call H5Dclose_f(dhnd,ret)
```

読込み

陸海フラグデータは、解像度毎に全周波数データがまとまって取得されます。
まとまったままでは扱い難いので、各周波数毎の配列に分割します。
格納値は 0～100 のフラグ値なので、各周波数毎に integer(1)型の 2 次元配列を用意します。

```
862       ! separate
863       do j=1,num+ovr*2
864         do i=1,AM2_DEF_SNUM_LO
865           lof06(i,j)=loflo(i,(num+ovr*2)*0+j)
866           lof07(i,j)=loflo(i,(num+ovr*2)*1+j)
867           lof10(i,j)=loflo(i,(num+ovr*2)*2+j)
              lof18(i,j)=loflo(i,(num+ovr*2)*3+j)
              lof23(i,j)=loflo(i,(num+ovr*2)*4+j)
              lof36(i,j)=loflo(i,(num+ovr*2)*5+j)
872       enddo
          enddo
          ! cutoff overlap
          do j=1,num
            do i=1,AM2_DEF_SNUM_LO
876           lof06(i,j)=lof06(i,j+ovr)
877           lof07(i,j)=lof07(i,j+ovr)
878           lof10(i,j)=lof10(i,j+ovr)
879           lof18(i,j)=lof18(i,j+ovr)
880           lof23(i,j)=lof23(i,j+ovr)
881           lof36(i,j)=lof36(i,j+ovr)
882         enddo
883       enddo
884       do j=num+1,LMT
885         lof06(:,j)=0
886         lof07(:,j)=0
887         lof10(:,j)=0
888         lof18(:,j)=0
889         lof23(:,j)=0
890         lof36(:,j)=0
891       enddo
```

各周波数に分解します

重複除去も行います

◇観測入射角データ読込み　※左側の数字は行番号です

観測入射角は格納型が 2byte 符号あり整数(-32768～32767)で、スケール値が設定されているので、
観測入射角に付属しているアトリビュートにアクセスして、スケール値を読み出します。
スケールを適用する際は、欠損値(-32768)と異常値(-32767)は除外します。

```
934 C read array: earth incidence
935       ! read
936       call H5Dopen_f(fhnd
937      +,'Earth Incidence',dhnd,ret)
938       call H5Aopen_f(dhnd,'SCALE FACTOR',ahnd,ret)     !
          call H5Aread_f(ahnd,H5T_NATIVE_REAL,sca,sz1,ret) !
          call H5Aclose_f(ahnd,ret)                        ! get scale
941       sz1(1)=LMT
942       sz1(2)=AM2_DEF_SNUM_LO
943       call H5Dread_f(dhnd
944      +,H5T_NATIVE_REAL,ear_in,sz1,ret,H5S_ALL_F,H5S_ALL_F)
945       if(ret.lt.0)then
946         write(*,'(a,a)')'H5Dread error: '
947      +  ,'Earth Incidence'
948         call exit(1)
949       endif
950       call H5Dclose_f(dhnd,ret)
951       ! cutoff overlap & change scale
952       do j=1,num
953         do i=1,AM2_DEF_SNUM_LO
954           ear_in(i,j)=ear_in(i,j+ovr)
955           if(ear_in(i,j).gt.-32767)ear_in(i,j)=ear_in(i,j)*sca
956         enddo
957       enddo
958       do j=num+1,LMT
959         ear_in(:,j)=0
960       enddo
961       ! sample display
962       write(*,'(a,f9.2)')'ear_in(pixel=1,scan=1): ',ear_in(1,1)
```

読込み

スケール読込み

重複除去

欠損値(-32768)と異常値(-32767)は除外して、
スケールを適用します。

◇終了処理　※左側の数字は行番号です

ファイルをクローズします。
HDF5 を終了します。

```
993       call H5Fclose_f(fhnd,ret)
994       call H5close_f(ret)
995       end
```

◇ファイルのクローズ
call H5Fclose_f(fhnd,ret)
fhnd:ファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇HDF5 の終了
call H5close_f(ret)
ret: [戻り値]失敗の場合は負の値

８．１．２ コンパイル方法(build_readL1B_hdf5_f.sh 解説)

コンパイルに使用するスクリプト build_readL1B_hdf5_f.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～8 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 8 SZIP=/home/user1/util/szip_2.1
 9
```

11～12 行目に、使用するコンパイラを指定します。
時刻変換と低周波緯度経度を算出する関数が C 言語のため、C コンパイラも指定します。
インテルコンパイラ(ifort/icc)または PG コンパイラ(pgf90/pgcc)を指定します。

```
10 # compiler
11 fc=ifort
12 cc=icc
13
14 # source filename
15 fsrc="readL1B_hdf5.f"
16 csrc="amsr2time_.c amsr2latlon_.c"
17 obj=`echo $csrc|sed "s/¥.c/.o/g"`
18
19 # output filename
20 out=readL1B_hdf5_f
21
22 # library order
23 lib="-lhdf5hl_fortran -lhdf5_hl -lhdf5_fortran -lhdf5 -lsz -lz -lm"
24
25 # c compile
26 cmd="$cc -g -c $csrc"
27 echo $cmd
28 $cmd
29
30 # f compile
31 cmd="$fc -g $fsrc $obj -o $out -I$HDF5/include -I$SZIP/include -L$HDF5/lib
-L$SZIP/lib $lib"
32 echo $cmd
33 $cmd
34
35 # garbage
36 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL1B_hdf5_f.sh
icc -g -c amsr2time_.c amsr2latlon_.c
ifort -g readL1B_hdf5.f amsr2time_.o amsr2latlon_.o -o readL1B_hdf5_f
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lhdf5hl_fortran -lhdf5_hl -lhdf5_fortran -lhdf5 -lsz -lz -lm
```

８．１．３ プログラム実行結果のサンプル

サンプルプログラムでは固定配列を多数使用しているため、環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞

```
$ unlimit
```

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、4 つとも実行してください。

```
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited
```

```
$ ./readL1B_hdf5_f GW1AM2_201207261145_055A_L1SGBTBR_0000000.h5
input file: GW1AM2_201207261145_055A_L1SGBTBR_0000000.h5
NumberOfScans(RETRIEVE BY ARRAY SIZE):          1979
OverlapScans(FIXED VALUE):            20
limit of NumberOfScans =   2200
CoRegistrationParameterA1(FIXED VALUE): 6G- 1.25000,7G- 1.00000,10G- 1.25000,18G- 1.25000,23G- 1.25000,36G- 1.00000
CoRegistrationParameterA2(FIXED VALUE): 6G- 0.00000,7G--0.10000,10G--0.25000,18G- 0.00000,23G--0.25000,36G- 0.00000
amsr2time: AMSR2_LEAP_DATA = /export/emc3/util/common/AMTK_AMSR2_DATA/leapsec.dat
amsr2time: year=1993 month= 7 tai93sec=   15638401.00
amsr2time: year=1994 month= 7 tai93sec=   47174402.00
amsr2time: year=1996 month= 1 tai93sec=   94608003.00
amsr2time: year=1997 month= 7 tai93sec=  141868804.00
amsr2time: year=1999 month= 1 tai93sec=  189302405.00
amsr2time: year=2006 month= 1 tai93sec=  410227206.00
amsr2time: year=2009 month= 1 tai93sec=  504921607.00
amsr2time: year=2012 month= 7 tai93sec=  615254408.00
amsr2time: number of leap second = 8
time(scan=1): 2012/07/26 11:45:43
latlon89a(pixel=1,scan=1): ( -73.3289, 136.7714)
latlon89b(pixel=1,scan=1): ( -73.4038, 137.1498)
latlonlm(pixel=1,scan=1): ( -73.3538, 136.6228)
latlon06(pixel=1,scan=1): ( -73.3592, 136.6213)
latlon07(pixel=1,scan=1): ( -73.3497, 136.6429)
latlon10(pixel=1,scan=1): ( -73.3506, 136.6001)
latlon18(pixel=1,scan=1): ( -73.3592, 136.6213)
latlon23(pixel=1,scan=1): ( -73.3506, 136.6001)
latlon36(pixel=1,scan=1): ( -73.3532, 136.6514)
tb06h(pixel=1,scan=1):     173.28
tb06v(pixel=1,scan=1):     208.22
tb07h(pixel=1,scan=1):     173.07
tb07v(pixel=1,scan=1):     207.54
tb10h(pixel=1,scan=1):     170.94
tb10v(pixel=1,scan=1):     204.95
tb18h(pixel=1,scan=1):     164.85
tb18v(pixel=1,scan=1):     199.84
tb23h(pixel=1,scan=1):     163.22
tb23v(pixel=1,scan=1):     196.53
tb36h(pixel=1,scan=1):     156.56
tb36v(pixel=1,scan=1):     186.39
tb89ah(pixel=1,scan=1):     163.76
tb89av(pixel=1,scan=1):     179.27
tb89bh(pixel=1,scan=1):     170.60
tb89bv(pixel=1,scan=1):     188.16
pdq06h(pixel=1,scan=1):    0
pdq06v(pixel=1,scan=1):    0
pdq07h(pixel=1,scan=1):    0
pdq07v(pixel=1,scan=1):    0
pdq10h(pixel=1,scan=1):    0
pdq10v(pixel=1,scan=1):    0
pdq18h(pixel=1,scan=1):    0
pdq18v(pixel=1,scan=1):    0
pdq23h(pixel=1,scan=1):    0
pdq23v(pixel=1,scan=1):    0
pdq36h(pixel=1,scan=1):    0
pdq36v(pixel=1,scan=1):    0
pdq89ah(pixel=1,scan=1):    0
pdq89av(pixel=1,scan=1):    0
pdq89bh(pixel=1,scan=1):    0
pdq89bv(pixel=1,scan=1):    0
lof06(pixel=1,scan=1): 100
lof07(pixel=1,scan=1): 100
lof10(pixel=1,scan=1): 100
lof18(pixel=1,scan=1): 100
lof23(pixel=1,scan=1): 100
lof36(pixel=1,scan=1): 100
lof89a(pixel=1,scan=1): 100
lof89b(pixel=1,scan=1): 100
ear_in(pixel=1,scan=1):      55.20
ear_az(pixel=1,scan=1):     144.76
```

８．２ L1R データ読込み

８．２．１ サンプルプログラム readL1R_hdf5.f 解説

サンプルプログラム readL1R_hdf5.f では、L1R データファイルから、以下の格納データを読込んで、89G A ホーン緯度経度から低周波緯度経度を算出し、内容をテキスト表示します。このサンプルプログラムでは、格納されている L1R 輝度温度の一部のみ取扱います。L1R 輝度温度データ一覧については、P.14「３．８　L1 リサンプリングデータ」を参照ください。

TAI93 形式時刻の変換には、サブルーチンを用意しています。

メタデータ	格納データ
※HDF5 の FORTRAN90 ライブラリでは可変長文字列メタデータを読込めません。 ＊ NumberOfScans は配列サイズから調べます。 以下は固定値を使用します。 ＊ OverlapScans	＊ Scan Time ＊ Latitude of Observation Point for 89A - Latitude of Observation Point for 89B - Longitude of Observation Point for 89A - Longitude of Observation Point for 89B ＊ Brightness Temperature (res06,6.9GHz,H) - Brightness Temperature (res06,6.9GHz,V) - Brightness Temperature (res06,7.3GHz,H) - Brightness Temperature (res06,7.3GHz,V) - Brightness Temperature (res10,10.7GHz,H) - Brightness Temperature (res10,10.7GHz,V) - Brightness Temperature (res23,18.7GHz,H) - Brightness Temperature (res23,18.7GHz,V) - Brightness Temperature (res23,23.8GHz,H) - Brightness Temperature (res23,23.8GHz,V) - Brightness Temperature (res36,36.5GHz,H) - Brightness Temperature (res36,36.5GHz,V) - Brightness Temperature (res36,89.0GHz,H) - Brightness Temperature (res36,89.0GHz,V) - Brightness Temperature (original,89GHz-A,H) - Brightness Temperature (original,89GHz-A,V) - Brightness Temperature (original,89GHz-B,H) - Brightness Temperature (original,89GHz-B,V) ＊ Pixel Data Quality 6 to 36 - Pixel Data Quality 89 ＊ Land_Ocean Flag 6 to 36 - Land_Ocean Flag 89 ＊ Earth Incidence - Earth Azimuth
89G A ホーン緯度経度から抽出するデータ	
＊ 低周波用緯度経度 → P.15 基礎知識編３．１０参照	

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の HDF5 関数は、最初に使用する際に使用説明を記載します。

HDF5 ライブラリでは、ファイルをオープンした後に、各データについてもオープンとクローズを繰り返します。各データの読込みは以下のような流れになります。

データをオープンする →必要な場合はスケールを取得する
→ データを読込む → データをクローズする

◇変数宣言　※左側の数字は行番号です

```
 1       program main
 2       use hdf5
 3       implicit none
 4 C include
 5       include 'amsr2time_f.h'
 :
 8 C fixed value
 9       integer(4),parameter::LMT=2200 ! limit of NumberOfScans
10       integer(4),parameter::AM2_DEF_SNUM_HI=486 ! high resolution pixel width
11       integer(4),parameter::AM2_DEF_SNUM_LO=243 ! low  resolution pixel width
12 C interface variable
13       integer(4) i,j          ! loop variable
14       integer(4) ret          ! return status
15       character(len=512) buf  ! text buffer
16       character(len=512) fn   ! filename
17       integer(HID_T) fhnd     ! file handle
18       integer(HID_T) ahnd     ! attribute handle
19       integer(HID_T) atyp     ! attribute type
20       integer(HID_T) dhnd     ! dataset handle
21       integer(HID_T) dtyp     ! dataset type
22       integer(HID_T) shnd     ! dataspace handle
23       integer(HSIZE_T) sz1(3) ! array size 1
24       integer(HSIZE_T) sz2(3) ! array size 2
 :
35       integer(4) num          ! NumberOfScans
36       integer(4) ovr          ! OverlapScans
 :
39       parameter(ovr=20)
 :
45       type(AM2_COMMON_SCANTIME) st(LMT) ! scantime
46       real(4) lat89ar(AM2_DEF_SNUM_HI,LMT) ! lat for 89a altitude revised
47       real(4) lat89br(AM2_DEF_SNUM_HI,LMT) ! lat for 89b altitu
 :
49       real(4) lon89ar(AM2_DEF_SNUM_HI,LMT) ! lon for 89a altitu
50       real(4) lon89br(AM2_DEF_SNUM_HI,LMT) ! lon for 89b altitude revised
 :
52       real(4) tb06h06(AM2_DEF_SNUM_LO,LMT) ! tb for 06h, resolution 06G
53       real(4) tb06v06(AM2_DEF_SNUM_LO,LMT) ! tb for 06v, resolution 06G
 :
70       integer(1) pdq06h(AM2_DEF_SNUM_LO,LMT)  ! pixel data quality for 06h
71       integer(1) pdq06v(AM2_DEF_SNUM_LO,LMT)  ! pixel data quality for 06v
 :
79       integer(1) lof06(AM2_DEF_SNUM_LO,LMT)   ! land ocean flag for 06
80       integer(1) lof10(AM2_DEF_SNUM_LO,LMT)   ! land ocean flag for 10
 :
86       real(4) ear_in(AM2_DEF_SNUM_LO,LMT) ! earth incidence
87       real(4) ear_az(AM2_DEF_SNUM_LO,LMT) ! earth azimuth
```

HDF5 用 use 文を宣言します。

時刻変換用サブルーチンヘッダーを
インクルードします。

標準プロダクトのスキャン数上限は
2200 で充分ですが、
準リアルプロダクトを使用する場合は、
2 周回程度繋がる場合があるので、
LMT=9000 としてください。
(標準の 4 パス分程度)

HDF5 用インターフェイス
変数を宣言します。

スキャン数
オーバーラップスキャン数

メタデータ用変数を宣言します。
※HDF5 の FORTRAN90 ライブラリでは
可変長文字列のメタデータを読込めな
いので、後で固定値を設定します。

時刻データには AM2_COMMON_SCANTIME 構造体を使います。
配列の次元数はデータによって異なります。
スキャン数はパスによって若干上下するので、上限を決めて宣言しておきます。
ここでは上限として LMT=2200 を設定しています。
AM2_DEF_SNUM_HI は高解像度ピクセル数(486)です。
AM2_DEF_SNUM_LO は低解像度ピクセル数(243)です。

時刻
緯度
経度
輝度温度
品質フラグ
陸海フラグ
観測入射角
観測方位角

配列データ用変数
を宣言します。

◇開始処理　※左側の数字は行番号です

HDF5 の初期化処理を行います。

◇HDF5 の初期化
call H5open_f(ret)
ret: [戻り値]失敗の場合は負の値

```
 96 C hdf5 initialize
 97       call H5open_f(ret)
 98       if(ret.lt.0)then
 99         write(*,'(a,i12)')'h5open_f error: ',ret
100         call exit(1)
101       endif
```

ファイルをオープンします。

◇HDF5 ファイルオープン
call H5Fopen_f(fn,label1,fhnd,ret,label2)
fn: オープンするファイル名を指定します
label1: アクセスモードを指定します H5F_ACC_RDONLY_F で読込専用になります
fhnd: [戻り値]開いたファイルのハンドル値
ret: [戻り値]失敗の場合は負の値
label2: H5P_DEFAULT_F を指定します

```
102 C open
103       call H5Fopen_f(fn,H5F_ACC_RDONLY_F,fhnd,ret,H5P_DEFAULT_F)
104       if(ret.lt.0)then
105         write(*,'(a,a)')'H5Fopen error: ',fn(1:len_trim(fn))
106         call exit(1)
107       endif
```

◇NumberOfScans の設定　※左側の数字は行番号です

HDF5 の FORTRAN90 ライブラリでは可変長文字列のメタデータを読み込めないので、
NumberOfScans を配列サイズから調べます。ここでは"Scan Time"のサイズを調べています。
配列サイズを取得するには、データセットをオープンして、データスペースをオープンしてから、
データセットのサイズ取得関数を使用します。不要になったハンドルはクローズします。

◇データセットのオープン
call H5Dopen_f(fhnd,nam,dhnd,ret)
fhnd: ファイルハンドル値を指定します
nam: データセット名を指定します
dhnd: [戻り値]データセットのハンドル値
ret: [戻り値]失敗の場合は負の値

◇データスペースのオープン
call H5Dget_space_f(dhnd,shnd,ret)
dhnd: データセットハンドル値を指定します
shnd: [戻り値]データスペースのハンドル値
ret: [戻り値]失敗の場合は負の値

◇データセットのサイズ取得
call H5Sget_simple_extent_dims_f(shnd,sz1,sz2,ret)
shnd: データスペースハンドル値を指定します
sz1: [戻り値]データセットの各次元サイズ
sz2: [戻り値]データセットの各次元の最大サイズ
ret: [戻り値]失敗の場合は負の値

```
108 C HDF5 FORTRAN LIBRARY CAN'T RETRIEVE VARIABLE STRING !
109 C calculate NumberOfScans from array size and OverlapScans
110       call H5Dopen_f(fhnd,'Scan Time',dhnd,ret)
111       call H5Dget_space_f(dhnd,shnd,ret)
112       call H5Sget_simple_extent_dims_f(shnd,sz1,sz2,ret)
113       if(ret.lt.0)then
114         write(*,'(a)')'H5Sget_simple_extent_dims error: Scan Time'
115         call exit(1)
116       endif
117       call H5Sclose_f(shnd,ret)
118       call H5Dclose_f(dhnd,ret)
119       num=sz1(1)-ovr*2
120       write(*,'(a,i12)')'NumberOfScans(RETRIEVE BY ARRAY SIZE): ',num
121       write(*,'(a,i12)')'OverlapScans(FIXED VALUE): ',ovr
```

NumberOfScans は、
「配列のスキャン数－(OverlapScans×２)」となります。

◇データスペースのクローズ
call H5Sclose_f(shnd,ret)
shnd: データスペースハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇データセットのクローズ
call H5Dclose_f(dhnd,ret)
dhnd: データセットハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇時刻データ読込み　※左側の数字は行番号です

データを読込むためには、データセットを H5Dopen()でオープンして、データタイプを取得して、
読込み用変数の各次元サイズを配列変数に設定しておきます。
L1 データには前後にオーバーラップが含まれるので重複部分を取り除きます。
時刻データの格納形式は TAI93 形式なので年/月/日/時/分/秒に変換します。

P.11 基礎知識編３． ５参照

◇データ読込み
call H5Dread_f(dhnd,dtyp,buf,sz1,ret,label1,label2)
dhnd: データセットハンドル値を指定します
dtyp: データタイプ値を指定します
buf: 読込み用の変数を指定します
sz1: buf の各次元サイズを設定しておきます
ret: [戻り値]失敗の場合は負の値
label1,label2: 部分配列を読込む場合に使用します
全て読込む場合は、どちらも H5S_ALL_F を指定します

```
153 C read array: scantime
154       ! read
155       call H5Dopen_f(fhnd,'Scan Time',dhnd,ret)
156       sz1(1)=LMT
157       call H5Dread_f(dhnd
          +,H5T_NATIVE_DOUBLE,r8d1,sz1,ret,H5S_ALL_F,H5S_ALL_F)
          if(ret.lt.0)then
160         write(*,'(a,a)')'H5Dread error: ','Scan Time'
161         call exit(1)
162       endif
163       call H5Dclose_f(dhnd,ret)
164       ! cutoff overlap
165       do j=1,num
166         r8d1(j)=r8d1(j+ovr)
          enddo
168       do j=num+1,LMT
169         r8d1(j)=0
170       enddo
171       ! convert
          call amsr2time(num,r8d1,st)
          ! sample display
174       write(*,'(a,i4.4,"/",i2.2,"/",i2.2," ",i2.2,":",i2.2,":",i2.2)')
175      +'time(scan=1): '
176      +,st(1)%year
177      +,st(1)%month
178      +,st(1)%day
179      +,st(1)%hour
180      +,st(1)%minute
181      +,st(1)%second
```

読込み

重複除去

時刻変換

AMSR2 プロダクトの時刻格納形式は、TAI93 形式と呼ばれる、
うるう秒を含めた 1993/01/01 からの通算秒です。
このままでは日時情報として扱い難いので、
このサンプルプログラムでは年/月/日/時/分/秒に変換する
サブルーチンを用意しています。この時刻変換は AMTK を使用
する場合は自動的に行われます。
→ P.11 基礎知識編３． ６参照

◆時刻形式の変換
call amsr2time(num,in,out)
num: スキャン数を指定します
in: TAI93 形式の時刻データを指定します
out: [戻り値]AM2_COMMON_SCANTIME 構造体で年/月/日/時/分/秒データが返されます

◇緯度経度データ読込み　※左側の数字は行番号です

89G 緯度経度は格納データを読込みます。
二次元配列を読込む場合には、サイズ指定配列の 1 番目と 2 番目に値を設定します。
読込んだ後は、オーバーラップを取り除きます。

P. 15 基礎知識編３． １０参照

低周波用緯度経度は、89G A ホーン緯度経度の 1 から数えて奇数番目を抽出します。

```
182 C read array: latlon for 89a altitude revised
183       ! read lat
184       call H5Dopen_f(fhnd
185      +,'Latitude of Observation Point for 89A',dhnd,ret)
186       sz1(1)=LMT
187       sz1(2)=AM2_DEF_SNUM_HI
188       call H5Dread_f(dhnd
189      +,H5T_NATIVE_REAL,lat89ar,sz1,ret,H5S_ALL_F,H5S_ALL_F)
190       if(ret.lt.0)then
191         write(*,'(a,a)')'H5Dread error: '
192      +  ,'Latitude of Observation Point for 89A'
193         call exit(1)
194       endif
195       call H5Dclose_f(dhnd,ret)
196       ! read lon
197       call H5Dopen_f(fhnd
198      +,'Longitude of Observation Point for 89A',dhnd,ret)
199       sz1(1)=LMT
200       sz1(2)=AM2_DEF_SNUM_HI
201       call H5Dread_f(dhnd
202      +,H5T_NATIVE_REAL,lon89ar,sz1,ret,H5S_ALL_F,H5S_ALL_F)
203       if(ret.lt.0)then
204         write(*,'(a,a)')'H5Dread error: '
205      +  ,'Longitude of Observation Point for 89A'
206         call exit(1)
207       endif
208       call H5Dclose_f(dhnd,ret)
209       ! cutoff overlap
210       do j=1,num
211         lat89ar(:,j)=lat89ar(:,j+ovr)
212         lon89ar(:,j)=lon89ar(:,j+ovr)
213       enddo
214       do j=num+1,LMT
215         lat89ar(:,j)=0
216         lon89ar(:,j)=0
217       enddo
 :

260 C read array: latlon for low resolution
261       do j=1,num
262         do i=1,AM2_DEF_SNUM_LO
263           latlr(i,j)=lat89ar(i*2-1,j)
264           lonlr(i,j)=lon89ar(i*2-1,j)
265         enddo
266       enddo
```

緯度読込み

経度読込み

重複除去

低周波抽出

◇輝度温度データ読込み　※左側の数字は行番号です

輝度温度は格納型が 2byte 符号なし整数(0～65535)で、スケール値が設定されているので、
輝度温度に付属しているアトリビュートにアクセスして、スケール値を読み出します。
スケールを適用する際は、欠損値(65535)と異常値(65534)は除外します。

◇アトリビュートのオープン
call H5Aopen_f(dhnd,nam,ahnd,ret)
dhnd: データセットハンドル値を指定します
nam: アトリビュート名を指定します
ahnd: [戻り値]アトリビュートのハンドル値
ret: [戻り値]失敗の場合は負の値

◇アトリビュート読込み
call H5Aread_f(ahnd,atyp,buf,sz1,ret)
ahnd: アトリビュートハンドル値を指定します
otyp: 出力変数の型を指定します
buf: 出力変数を指定します
sz1: buf の各次元サイズを設定しておきます(スカラ読込みでは無視されます)
ret: [戻り値]失敗の場合は負の値

```
269 C read array: tb for 06h, resolution 06G
270       ! read
271       call H5Dopen_f(fhnd
27       +,'Brightness Temperature (res06,6.9GHz,H)',dhnd,ret)
27        call H5Aopen_f(dhnd,'SCALE FACTOR',ahnd,ret)     ! g
274       call H5Aread_f(ahnd,H5T_NATIVE_REAL,sca,sz1,ret) !
275       call H5Aclose_f(ahnd,ret)                        ! get scale
276       sz1(1)=LMT
277       sz1(2)=AM2_DEF_SNUM_LO
278       call H5Dread_f(dhnd
         +,H5T_NATIVE_REAL,tb06h06,sz1,ret,H5S_ALL_F,H5S_ALL_F)
          if(ret.lt.0)then
281         write(*,'(a,a)')'H5Dread error: '
282       +  ,'Brightness Temperature (res06,6.9GHz,H)'
28          call exit(1)
284       endif
285       call H5Dclose_f(dhnd,ret)
286       ! cutoff overlap & convert to unsignd & change scale
287       do j=1,num
28          do i=1,AM2_DEF_SNUM_LO
289           tb06h06(i,j)=tb06h06(i,j+ovr)
290           if(tb06h06(i,j).lt.65534)tb06h06(i,j)=tb06h06(i,j)*sca
291         enddo
          enddo
          do j=num+1,LMT
29          tb06h06(:,j)=0
295       enddo
296       ! sample display
297       write(*,'(a,f9.2)')'tb06h06(pixel=1,scan=1): ',tb06h06(1,1)
```

スケール読込み

読込み

重複除去

欠損値(65535)と異常値(65534)は除外して、
スケールを適用します。

◇アトリビュートのクローズ
call H5Aclose_f(ahnd,ret)
ahnd:アトリビュートハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

このスケール処理は AMTK を使用する場合は
自動的に行われます。
→ P.16 基礎知識編３． １２参照

◇L1 品質フラグデータ読込み　※左側の数字は行番号です

L1 低周波品質フラグは 2 つの 1byte 整数型で 16 ビットの内 12 ビットが各低周波に対応します。
L1 高周波品質フラグは 1 つの 1byte 整数型で 8 ビットの内 4 ビットが各高周波に対応します。

読込み

```
791 C read array: pixel data quality for low
792       ! read
79        call H5Dopen_f(fhnd
79       +,'Pixel Data Quality 6 to 36',dhnd,ret)
795       sz1(1)=LMT
796       sz1(2)=AM2_DEF_SNUM_LO
          call H5Dread_f(dhnd
         +,H5T_NATIVE_INTEGER,i4d2hi,sz1,ret,H5S_ALL_F,H5S_ALL_F)
799       if(ret.lt.0)then
800         write(*,'(a,a)')'H5Dread error: '
801        +  ,'Pixel Data Quality 6 to 36'
80          call exit(1)
80        endif
804       call H5Dclose_f(dhnd,ret)
```

L1 品質フラグデータは、各周波数（6GHz、7GHz）、偏波に対する RFI 情報がまとまって取得されます。
まとまったままでは扱い難いので、各周波数毎の配列に分割します。
格納値は 2 ビットのマスク値なので、各周波数毎に integer(1)型の 2 次元配列を用意します。
L1 品質フラグデータには RFI(Radio Frequency Interference；電波干渉)情報が格納されています。
電波干渉の発生がない場合は 00、発生の可能性がある場合は 10、発生がある場合は 11 となります。

ビット毎の情報を各周波数に分解

重複除去も行います

```
805       ! cutoff overlap & separate
806       do j=1,num
807         do i=1,AM2_DEF_SNUM_LO
808          pdq06v(i,j)=0
80           if(btest(i4d2hi((i-1)*2+1,j+ovr),0))pdq06v(i,j)=1
81           if(btest(i4d2hi((i-1)*2+1,j+ovr),1))pdq06v(i,j)=10
             if(btest(i4d2hi((i-1)*2+1,j+ovr),0)
            +    .and.btest(i4d2hi((i-1)*2+1,j+ovr),1))pdq06v(i,j)=11
             pdq06h(i,j)=0
814          if(btest(i4d2hi((i-1)*2+1,j+ovr),2))pdq06h(i,j)=1
             if(btest(i4d2hi((i-1)*2+1,j+ovr),3))pdq06h(i,j)=10
             if(btest(i4d2hi((i-1)*2+1,j+ovr),2)
            +    .and.btest(i4d2hi((i-1)*2+1,j+ovr),3))pdq06h(i,j)=11
818          pdq07v(i,j)=0
819          if(btest(i4d2hi((i-1)*2+1,j+ovr),4))pdq07v(i,j)=1
820          if(btest(i4d2hi((i-1)*2+1,j+ovr),5))pdq07v(i,j)=10
821          if(btest(i4d2hi((i-1)*2+1,j+ovr),4)
822         +    .and.btest(i4d2hi((i-1)*2+1,j+ovr),5))pdq07v(i,j)=11
823          pdq07h(i,j)=0
824          if(btest(i4d2hi((i-1)*2+1,j+ovr),6))pdq07h(i,j)=1
825          if(btest(i4d2hi((i-1)*2+1,j+ovr),7))pdq07h(i,j)=10
826          if(btest(i4d2hi((i-1)*2+1,j+ovr),6)
827         +    .and.btest(i4d2hi((i-1)*2+1,j+ovr),7))pdq07h(i,j)=11
828        enddo
82        enddo
83        do j=num+1,LMT
831         pdq06h(:,j)=0
832         pdq06v(:,j)=0
833         pdq07h(:,j)=0
834         pdq07v(:,j)=0
835       enddo
```

◇L1R 陸海フラグデータ読込み　※左側の数字は行番号です

L1R 陸海は integer(1)型の 2 次元配列((スキャン数*チャンネル数)×ピクセル数)です。
低周波データは解像度が 4 チャンネル(6G/10G/23G/36G)あります。
高周波データは解像度が 2 チャンネル(89G A ホーン/89G B ホーン)あります。

P. 14 基礎知識編３． ９参照

```
875 C read array: land ocean flag for low
876       ! read
877       call H5Dopen_f(fhnd
87       +,'Land_Ocean Flag 6 to 36',dhnd,ret)
879       sz1(1)=LMT*6
880       sz1(2)=AM2_DEF_SNUM_LO
          call H5Dread_f(dhnd
         +,H5T_NATIVE_INTEGER,loflo,sz1,ret,H5S_ALL_F,H5S_ALL_F)
883       if(ret.lt.0)then
884         write(*,'(a,a)')'H5Dread error: '
885       +  ,'Land_Ocean Flag 6 to 36'
886         call exit(1)
887       endif
888       call H5Dclose_f(dhnd,ret)
```

読込み

陸海フラグデータは、解像度毎に全周波数データがまとまって取得されます。
まとまったままでは扱い難いので、各周波数毎の配列に分割します。
格納値は 0～100 のフラグ値なので、各周波数毎に integer(1)型の 2 次元配列を用意します。

```
889       ! separate
890       do j=1,num+ovr*2
891         do i=1,AM2_DEF_SNUM_LO
892           lof06(i,j)=loflo(i,(num+ovr*2)*0+j)
893           lof10(i,j)=loflo(i,(num+ovr*2)*1+j)
894           lof23(i,j)=loflo(i,(num+ovr*2)*2+j)
              lof36(i,j)=loflo(i,(num+ovr*2)*3+j)
          enddo
        enddo
        ! cutoff overlap
899     do j=1,num
          do i=1,AM2_DEF_SNUM_LO
            lof06(i,j)=lof06(i,j+ovr)
            lof10(i,j)=lof10(i,j+ovr)
903         lof23(i,j)=lof23(i,j+ovr)
904         lof36(i,j)=lof36(i,j+ovr)
905       enddo
906     enddo
907     do j=num+1,LMT
908       lof06(:,j)=0
909       lof10(:,j)=0
910       lof23(:,j)=0
911       lof36(:,j)=0
912     enddo
```

各周波数に分解します

重複除去も行います

◇観測入射角データ読込み　※左側の数字は行番号です

観測入射角は格納型が 2byte 符号あり整数(-32768～32767)で、スケール値が設定されているので、
観測入射角に付属しているアトリビュートにアクセスして、スケール値を読み出します。
スケールを適用する際は、欠損値(-32768)と異常値(-32767)は除外します。

```
953 C read array: earth incidence
954       ! read
955       call H5Dopen_f(fhnd
956      +,'Earth Incidence',dhnd,ret)
957       call H5Aopen_f(dhnd,'SCALE FACTOR',ahnd,ret)     !
          call H5Aread_f(ahnd,H5T_NATIVE_REAL,sca,sz1,ret) !
          call H5Aclose_f(ahnd,ret)                        ! get scale
960       sz1(1)=LMT
961       sz1(2)=AM2_DEF_SNUM_LO
962       call H5Dread_f(dhnd
963      +,H5T_NATIVE_REAL,ear_in,sz1,ret,H5S_ALL_F,H5S_ALL_F)
964       if(ret.lt.0)then
965         write(*,'(a,a)')'H5Dread error: '
966      +  ,'Earth Incidence'
967         call exit(1)
968       endif
969       call H5Dclose_f(dhnd,ret)
970       ! cutoff overlap & change scale
971       do j=1,num
972         do i=1,AM2_DEF_SNUM_LO
974           ear_in(i,j)=ear_in(i,j+ovr)
              if(ear_in(i,j).gt.-32767)ear_in(i,j)=ear_in(i,j)*sca
975       enddo
976       enddo
977       do j=num+1,LMT
978         ear_in(:,j)=0
979       enddo
980       ! sample display
981       write(*,'(a,f9.2)')'ear_in(pixel=1,scan=1): ',ear_in(1,1)
```

読込み

スケール読込み

重複除去

欠損値(-32768)と異常値(-32767)は除外して、
スケールを適用します。

◇終了処理　※左側の数字は行番号です

ファイルをクローズします。
HDF5 を終了します。

```
1012      call H5Fclose_f(fhnd,ret)
1013      call H5close_f(ret)
1014      end
```

◇ファイルのクローズ
call H5Fclose_f(fhnd,ret)
fhnd:ファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇HDF5 の終了
call H5close_f(ret)
ret: [戻り値]失敗の場合は負の値

８．２．２ コンパイル方法(build_readL1R_hdf5_f.sh 解説)

コンパイルに使用するスクリプト build_readL1R_hdf5_f.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～8 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 8 SZIP=/home/user1/util/szip_2.1
 9
```

11～12 行目に、使用するコンパイラを指定します。
時刻変換と低周波緯度経度を算出する関数が C 言語のため、C コンパイラも指定します。
インテルコンパイラ(ifort/icc)または PG コンパイラ(pgf90/pgcc)を指定します。

```
10 # compiler
11 fc=ifort
12 cc=icc
13
14 # source filename
15 fsrc="readL1R_hdf5.f"
16 csrc="amsr2time_.c"
17 obj=`echo $csrc|sed "s/¥.c/.o/g"`
18
19 # output filename
20 out=readL1R_hdf5_f
21
22 # library order
23 lib="-lhdf5hl_fortran -lhdf5_hl -lhdf5_fortran -lhdf5 -lsz -lz -lm"
24
25 # c compile
26 cmd="$cc -g -c $csrc"
27 echo $cmd
28 $cmd
29
30 # f compile
31 cmd="$fc -g $fsrc $obj -o $out -I$HDF5/include -I$SZIP/include -L$HDF5/lib
-L$SZIP/lib $lib"
32 echo $cmd
33 $cmd
34
35 # garbage
36 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL1R_hdf5_f.sh
icc -g -c amsr2time_.c
ifort -g readL1R_hdf5.f amsr2time_.o -o readL1R_hdf5_f
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lhdf5hl_fortran -lhdf5_hl -lhdf5_fortran -lhdf5 -lsz -lz -lm
```

８．２．３ プログラム実行結果のサンプル

サンプルプログラムでは固定配列を多数使用しているため、環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞
```
$ unlimit
```

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、4 つとも実行してください。
```
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited
```

```
$ ./readL1R_hdf5_f GW1AM2_201207261145_055A_L1SGRTBR_
0000000.h5
input file: GW1AM2_201207261145_055A_L1SGRTBR_000000
0.h5
NumberOfScans(RETRIEVE BY ARRAY SIZE):          1979
OverlapScans(FIXED VALUE):            20
limit of NumberOfScans =   2200
amsr2time: AMSR2_LEAP_DATA = /export/emc3/util/common
/AMTK_AMSR2_DATA/leapsec.dat
amsr2time: year=1993 month= 7 tai93sec=   15638401.00
amsr2time: year=1994 month= 7 tai93sec=   47174402.00
amsr2time: year=1996 month= 1 tai93sec=   94608003.00
amsr2time: year=1997 month= 7 tai93sec=  141868804.00
amsr2time: year=1999 month= 1 tai93sec=  189302405.00
amsr2time: year=2006 month= 1 tai93sec=  410227206.00
amsr2time: year=2009 month= 1 tai93sec=  504921607.00
amsr2time: year=2012 month= 7 tai93sec=  615254408.00
amsr2time: number of leap second = 8
time(scan=1): 2012/07/26 11:45:43
latlon89ar(pixel=1,scan=1): ( -73.3581, 136.8432)
latlon89br(pixel=1,scan=1): ( -73.4328, 137.2216)
latlonlr(pixel=1,scan=1): ( -73.3581, 136.8432)
tb06h06(pixel=1,scan=1):    173.41
tb06v06(pixel=1,scan=1):    208.09
tb07h06(pixel=1,scan=1):    173.07
tb07v06(pixel=1,scan=1):    207.11
tb10h10(pixel=1,scan=1):    170.40
tb10v10(pixel=1,scan=1):    204.58
tb18h23(pixel=1,scan=1):    165.83
tb18v23(pixel=1,scan=1):    199.41
tb23h23(pixel=1,scan=1):    163.55
tb23v23(pixel=1,scan=1):    195.90
```

```
tb36h36(pixel=1,scan=1):    153.55
tb36v36(pixel=1,scan=1):    183.95
tb89h36(pixel=1,scan=1):    163.86
tb89v36(pixel=1,scan=1):    181.05
tb89ah(pixel=1,scan=1):   163.76
tb89av(pixel=1,scan=1):   179.27
tb89bh(pixel=1,scan=1):   170.60
tb89bv(pixel=1,scan=1):   188.16
pdq06h(pixel=1,scan=1):   0
pdq06v(pixel=1,scan=1):   0
pdq07h(pixel=1,scan=1):   0
pdq07v(pixel=1,scan=1):   0
pdq10h(pixel=1,scan=1):   0
pdq10v(pixel=1,scan=1):   0
pdq18h(pixel=1,scan=1):   0
pdq18v(pixel=1,scan=1):   0
pdq23h(pixel=1,scan=1):   0
pdq23v(pixel=1,scan=1):   0
pdq36h(pixel=1,scan=1):   0
pdq36v(pixel=1,scan=1):   0
pdq89ah(pixel=1,scan=1):   0
pdq89av(pixel=1,scan=1):   0
pdq89bh(pixel=1,scan=1):   0
pdq89bv(pixel=1,scan=1):   0
lof06(pixel=1,scan=1): 100
lof10(pixel=1,scan=1): 100
lof23(pixel=1,scan=1): 100
lof36(pixel=1,scan=1): 100
lof89a(pixel=1,scan=1): 100
lof89b(pixel=1,scan=1): 100
ear_in(pixel=1,scan=1):     55.20
ear_az(pixel=1,scan=1):    144.76
```

８．３ L2 低解像度データ読込み

積算雲水量(CLW)・海氷密接度(SIC)・土壌水分量(SMC)・積雪深(SND)・海面水温(SST)・海上風速(SSW)・可降水量(TPW)が L2 低解像度です。降水量(PRC)は L2 高解像度を参照ください。

８．３．１ サンプルプログラム readL2L_hdf5.f 解説

サンプルプログラム readL2L_hdf5.f では、L2 低解像度データファイルから、以下の格納データを読込んで、内容をテキスト表示します。
TAI93 形式時刻の変換には、サブルーチンを用意しています。

メタデータ	格納データ
※HDF5 の FORTRAN90 ライブラリでは可変長文字列メタデータを読込めません。 ＊ NumberOfScans は配列サイズから調べます。 以下は固定値を使用します。 ＊ OverlapScans	＊ Scan Time ＊ Latitude of Observation Point ＊ Longitude of Observation Point ＊ Geophysical Data ＊ Pixel Data Quality P.15 基礎知識編３．１０参照

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の HDF5 関数は、最初に使用する際に使用説明を記載します。

HDF5 ライブラリでは、ファイルをオープンした後に、各データについてもオープンとクローズを繰り返します。各データの読込みは以下のような流れになります。

データをオープンする →必要な場合はスケールを取得する
→ データを読込む → データをクローズする

◇変数宣言　※左側の数字は行番号です

```
 1      program main
 2      use hdf5
 3      implicit none
 4 C include
 5      include 'amsr2time_f.h'
 :
 8 C fixed value
 9      integer(4),parameter::LMT=2200 ! limit of NumberOfScans
10      integer(4),parameter::AM2_DEF_SNUM_HI=486 ! high resolution pixel width
11      integer(4),parameter::AM2_DEF_SNUM_LO=243 ! low  resolution pixel width
 :
15 C interface variable
16      integer(4) i,j          ! loop variable
17      integer(4) ret          ! return status
18      character(len=512) buf  ! text buffer
19      character(len=512) fn   ! filename
20      integer(HID_T) fhnd     ! file handle
21      integer(HID_T) ahnd     ! attribute handle
22      integer(HID_T) atyp     ! attribute type
23      integer(HID_T) dhnd     ! dataset handle
24      integer(HID_T) shnd     ! dataspace handle
25      integer(HSIZE_T) sz1(3) ! array size 1
26      integer(HSIZE_T) sz2(3) ! array size 2
 :
37      integer(4) num          ! NumberOfScans
38      integer(4) ovr          ! OverlapScans
39      parameter(ovr=0)
 :
43 C array data
44      type(AM2_COMMON_SCANTIME) st(LMT) ! scantime
45      real(4) lat(AM2_DEF_SNUM_LO,LMT)  ! lat
46      real(4) lon(AM2_DEF_SNUM_LO,LMT)  ! lon
47      real(4) geo1(AM2_DEF_SNUM_LO,LMT) ! geophysical data layer 1
48      real(4) geo2(AM2_DEF_SNUM_LO,LMT) ! geophysical data layer 2
49      real(4) geotmp(AM2_DEF_SNUM_LO,LMT*2) ! geophysical data temporary
50      integer(4) pdq1(AM2_DEF_SNUM_LO,LMT) ! pixel data quality layer 1
51      integer(4) pdq2(AM2_DEF_SNUM_LO,LMT) ! pixel data quality layer 2
52      integer(4) pdqtmp(AM2_DEF_SNUM_LO,LMT*2) ! pixel data quality temporary
```

HDF5 用 use 文を宣言します。

時刻変換用サブルーチンヘッダーを
インクルードします。

標準プロダクトのスキャン数上限は
2200 で充分ですが、
準リアルプロダクトを使用する場合は、
2 周回程度繋がる場合があるので、
LMT=9000 としてください。
(標準の 4 パス分程度)

HDF5 用インターフェイス
変数を宣言します。

スキャン数
オーバーラップスキャン数

メタデータ用変数を宣言します。
※HDF5 の FORTRAN90 ライブラリでは
可変長文字列のメタデータを読込めな
いので、後で固定値を設定します。

時刻データには AM2_COMMON_SCANTIME 構造体を使います。
配列の次元数はデータによって異なります。
スキャン数はパスによって若干上下するので、上限を決めて宣言しておきます。
ここでは上限として LMT=2200 を設定しています。
AM2_DEF_SNUM_HI は高解像度ピクセル数(486)です。
AM2_DEF_SNUM_LO は低解像度ピクセル数(243)です。

時刻
緯度
経度
物理量
品質フラグ

配列データ用変数
を宣言します。

◇開始処理　※左側の数字は行番号です

HDF5 の初期化処理を行います。

◇HDF5 の初期化
call H5open_f(ret)
ret: [戻り値]失敗の場合は負の値

```
61 C hdf5 initialize
62       call H5open_f(ret)
63       if(ret.lt.0)then
64         write(*,'(a,i12)')'h5open_f error: ',ret
65         call exit(1)
66       endif
```

ファイルをオープンします。

◇HDF5 ファイルオープン
call H5Fopen_f(fn,label1,fhnd,ret,label2)
fn: オープンするファイル名を指定します
label1: アクセスモードを指定します H5F_ACC_RDONLY_F で読込専用になります
fhnd: [戻り値]開いたファイルのハンドル値
ret: [戻り値]失敗の場合は負の値
label2: H5P_DEFAULT_F を指定します

```
67 C open
68       call H5Fopen_f(fn,H5F_ACC_RDONLY_F,fhnd,ret,H5P_DEFAULT_F)
69       if(ret.lt.0)then
70         write(*,'(a,a)')'H5Fopen error: ',fn(1:len_trim(fn))
71         call exit(1)
72       endif
```

◇NumberOfScans の設定　※左側の数字は行番号です

HDF5 の FORTRAN90 ライブラリでは可変長文字列のメタデータを読み込めないので、
NumberOfScans を配列サイズから調べます。ここでは"Scan Time"のサイズを調べています。
配列サイズを取得するには、データセットをオープンして、データスペースをオープンしてから、
データセットのサイズ取得関数を使用します。不要になったハンドルはクローズします。

◇データセットのオープン
call H5Dopen_f(fhnd, nam, dhnd, ret)
fhnd: ファイルハンドル値を指定します
nam: データセット名を指定します
dhnd: [戻り値]データセットのハンドル値
ret: [戻り値]失敗の場合は負の値

◇データスペースのオープン
call H5Dget_space_f(dhnd, shnd, ret)
dhnd: データセットハンドル値を指定します
shnd: [戻り値]データスペースのハンドル値
ret: [戻り値]失敗の場合は負の値

◇データセットのサイズ取得
call H5Sget_simple_extent_dims_f(shnd, sz1, sz2, ret)
shnd: データスペースハンドル値を指定します
sz1: [戻り値]データセットの各次元サイズ
sz2: [戻り値]データセットの各次元の最大サイズ
ret: [戻り値]失敗の場合は負の値

```
 90 C HDF5 FORTRAN LIBRARY CAN'T RETRIEVE VARIABLE STRING !
 91 C calculate NumberOfScans from array size and OverlapScans
 92       call H5Dopen_f(fhnd,'Scan Time',dhnd,ret)
 93       call H5Dget_space_f(dhnd,shnd,ret)
 94       call H5Sget_simple_extent_dims_f(shnd,sz1,sz2,ret)
 95       if(ret.lt.0)then
 96         write(*,'(a)')'H5Sget_simple_extent_dims error: Scan Time'
 97         call exit(1)
 98       endif
 99       call H5Sclose_f(shnd,ret)
100       call H5Dclose_f(dhnd,ret)
101       num=sz1(1)-ovr*2
102       write(*,'(a,i12)')'NumberOfScans(RETRIEVE BY ARRAY SIZE): ',num
103       write(*,'(a,i12)')'OverlapScans(FIXED VALUE): ',ovr
```

NumberOfScans は、
「配列のスキャン数－(OverlapScans×２)」となります。

◇データスペースのクローズ
call H5Sclose_f(shnd, ret)
shnd: データスペースハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇データセットのクローズ
call H5Dclose_f(dhnd, ret)
dhnd: データセットハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇時刻データ読込み　※左側の数字は行番号です

データを読込むためには、データセットを H5Dopen()でオープンして、読込み用変数の各次元サイズを配列変数に設定しておきます。
時刻データの格納形式は TAI93 形式なので年/月/日/時/分/秒に変換します。

◇データ読込み
call H5Dread_f(dhnd,dtyp,buf,sz1,ret,label1,label2)
dhnd: データセットハンドル値を指定します
dtyp: データタイプ値を指定します
buf: 読込み用の変数を指定します
sz1: buf の各次元サイズを設定しておきます
ret: [戻り値]失敗の場合は負の値
label1,label2: 部分配列を読込む場合に使用します
全て読込む場合は、どちらも H5S_ALL_F を指定します

```
111 C read array: scantime
112       ! read
113       call H5Dopen_f(fhnd,'Scan Time',dhnd,ret)
114       sz1(1)=LMT
          call H5Dread_f(dhnd
         +,H5T_NATIVE_DOUBLE,r8d1,sz1,ret,H5S_ALL_F,H5S_ALL_F)
117       if(ret.lt.0)then
118         write(*,'(a,a)')'H5Dread error: ','Scan Time'
119         call exit(1)
120       endif
121       call H5Dclose_f(dhnd,ret)
122       ! convert
          call amsr2time(num,r8d1,st)
          ! sample display
125       write(*,'(a,i4.4,"/",i2.2,"/",i2.2," ",i2.2,":",i2.2,":",i2.2)')
126      +'time(scan=1): '
127      +,st(1)%year
128      +,st(1)%month
129      +,st(1)%day
130      +,st(1)%hour
131      +,st(1)%minute
132      +,st(1)%second
```

読込み

時刻変換

AMSR2 プロダクトの時刻格納形式は、TAI93 形式と呼ばれる、うるう秒を含めた 1993/01/01 からの通算秒です。
このままでは日時情報として扱い難いので、
このサンプルプログラムでは年/月/日/時/分/秒に変換するサブルーチンを用意しています。この時刻変換は AMTK を使用する場合は自動的に行われます。
→ P.11 基礎知識編３．６参照

◆時刻形式の変換
call amsr2time(num,in,out)
num: スキャン数を指定します
in: TAI93 形式の時刻データを指定します
out: [戻り値]AM2_COMMON_SCANTIME 構造体で年/月/日/時/分/秒データが返されます

◇緯度経度データ読込み　※左側の数字は行番号です

緯度経度データを読込みます。 → P.15 基礎知識編３．１０参照

緯度読込み

経度読込み

```
133 C read array: latlon
134       ! read lat
135       call H5Dopen_f(fhnd,'Latitude of Observation Point',dhnd,ret)
136       sz1(1)=LMT
137       sz1(2)=AM2_DEF_SNUM_LO
138       call H5Dread_f(dhnd
139      +,H5T_NATIVE_REAL,lat,sz1,ret,H5S_ALL_F,H5S_ALL_F)
140       if(ret.lt.0)then
141         write(*,'(a,a)')'H5Dread error: '
142      +  ,'Latitude of Observation Point'
143         call exit(1)
144       endif
145       call H5Dclose_f(dhnd,ret)
146       ! read lon
147       call H5Dopen_f(fhnd,'Longitude of Observation Point',dhnd,ret)
148       sz1(1)=LMT
149       sz1(2)=AM2_DEF_SNUM_LO
150       call H5Dread_f(dhnd
151      +,H5T_NATIVE_REAL,lon,sz1,ret,H5S_ALL_F,H5S_ALL_F)
152       if(ret.lt.0)then
153         write(*,'(a,a)')'H5Dread error: '
154      +  ,'Longitude of Observation Point'
155         call exit(1)
156       endif
157       call H5Dclose_f(dhnd,ret)
158       ! sample display
159       write(*,'(a,"(",f9.4,",",f9.4,")")')'latlon(pixel=1,scan=1): '
160      +,lat(1,1),lon(1,1)
```

◇物理量データ読込み　※左側の数字は行番号です

物理量データを読込みます。
積雪深(SND)・海面水温(SST)は物理量が 2 層あるので、この 2 つとそれ以外で処理を分けています。
積雪深の 2 層目は積雪水量[cm]です。海面水温の 2 層目は 10GHz による SST[℃]です。

物理量は格納型が 2byte 符号あり整数(-32768～32767)で、スケール値が設定されているので、
物理量に付属しているアトリビュートにアクセスして、スケール値を読み出します。
スケールを適用する際は、欠損値(-32768)と異常値(-32767)は除外します。

◇アトリビュートのオープン
call H5Aopen_f(dhnd,nam,ahnd,ret)
dhnd: データセットハンドル値を指定します
nam: アトリビュート名を指定します
ahnd: [戻り値]アトリビュートのハンドル値
ret: [戻り値]失敗の場合は負の値

◇アトリビュートのクローズ
call H5Aclose_f(ahnd,ret)
ahnd:アトリビュートハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇アトリビュート読込み
call H5Aread_f(ahnd,atyp,buf,sz1,ret)
ahnd: アトリビュートハンドル値を指定します
otyp: 出力変数の型を指定します
buf: 出力変数を指定します
sz1: buf の各次元サイズを設定しておきます(スカラ読込みでは無視されます)
ret: [戻り値]失敗の場合は負の値

```
161 C read array: geophysical data for 1 layer
162       if((gid(30:32).ne.'SND').and.(gid(30:32).ne.'SST'))then
163         call H5Dopen_f(fhnd,'Geophysical Data',dhnd,ret)
164         ! get scale
165         call H5Aopen_f(dhnd,'SCALE FACTOR',ahnd,ret)
166         call H5Aread_f(ahnd,H5T_NATIVE_REAL,sca,sz1,ret)
167         call H5Aclose_f(ahnd,ret)
168         ! read
169         sz1(1)=LMT
170         sz1(2)=AM2_DEF_SNUM_LO
171         call H5Dread_f(dhnd
172      +  ,H5T_NATIVE_REAL,geo1,sz1,ret,H5S_ALL_F,H5S_ALL_F)
173         if(ret.lt.0)then
174           write(*,'(a,a)')'H5Dread error: '
175      +    ,'Geophysical Data'
176           call exit(1)
177         endif
178         call H5Dclose_f(dhnd,ret)
179         ! change scale
180         do j=1,num
181           do i=1,AM2_DEF_SNUM_LO
182             if(geo1(i,j).gt.-32767)geo1(i,j)=geo1(i,j)*sca
183           enddo
184         enddo
185       endif
```

スケール読込み

読込み

スケール処理

このスケール処理は AMTK を使用する場合は
自動的に行われます。
→ P.16 基礎知識編３． １２参照

欠損値(-32768)と異常値(-32767)は除外して、
スケールを適用します。

2 層ある場合は、一時変数で全体を読込んだ後に、層ごとに分割しています。

```
186 C read array: geophysical data for 2 layer
187      if((gid(30:32).eq.'SND').or.(gid(30:32).eq.'SST'))then
188        call H5Dopen_f(fhnd,'Geophysical Data',dhnd,ret)
189        ! get scale
190        call H5Aopen_f(dhnd,'SCALE FACTOR',ahnd,ret)
191        call H5Aread_f(ahnd,H5T_NATIVE_REAL,sca,sz1,ret)
192        call H5Aclose_f(ahnd,ret)
193        ! read
194        sz1(1)=LMT*2
195        sz1(2)=AM2_DEF_SNUM_LO
196        call H5Dread_f(dhnd
197     +  ,H5T_NATIVE_REAL,geotmp,sz1,ret,H5S_ALL_F,H5S_ALL_F)
198        if(ret.lt.0)then
199          write(*,'(a,a)')'H5Dread error: '
200     +    ,'Geophysical Data'
201          call exit(1)
202        endif
203        call H5Dclose_f(dhnd,ret)
204        ! separate & change scale
205        do j=1,num
206          do i=1,AM2_DEF_SNUM_LO
207            geo1(i,j)=geotmp((i-1)*2+1,(j-1)*2+1)
208            geo2(i,j)=geotmp((i-1)*2+2,(j-1)*2+1)
209            if(geo1(i,j).gt.-32767)geo1(i,j)=geo1(i,j)*sca
210            if(geo2(i,j).gt.-32767)geo2(i,j)=geo2(i,j)*sca
211          enddo
212        enddo
213      endif
```

◇L2 品質フラグデータ読込み　※左側の数字は行番号です

L2 品質フラグデータを読込みます。
積雪深(SND)、海面水温(SST)は品質フラグが 2 層あるのでこの 2 つとそれ以外で処理を分けています。

```
214 C read array: pixel data quality for 1 layer
215       if((gid(30:32).ne.'SND').and.(gid(30:32).ne.'SST'))then
216         call H5Dopen_f(fhnd,'Pixel Data Quality',dhnd,ret)
217         sz1(1)=LMT
218         sz1(2)=AM2_DEF_SNUM_LO
219         call H5Dread_f(dhnd
220      +  ,H5T_NATIVE_INTEGER,pdq1,sz1,ret,H5S_ALL_F,H5S_ALL_F)
221         if(ret.lt.0)then
222           write(*,'(a,a)')'H5Dread error: '
223      +    ,'Pixel Data Quality'
224           call exit(1)
225         endif
226         call H5Dclose_f(dhnd,ret)
227       endif
```

2 層ある場合は、一時変数で全体を読込んだ後に、層ごとに分割しています。

```
228 C read array: pixel data quality for 2 layer
229       if((gid(30:32).eq.'SND').or.(gid(30:32).eq.'SST'))then
230         ! read
231         call H5Dopen_f(fhnd,'Pixel Data Quality',dhnd,ret)
232         sz1(1)=LMT*2
233         sz1(2)=AM2_DEF_SNUM_LO
234         call H5Dread_f(dhnd
235      +  ,H5T_NATIVE_INTEGER,pdqtmp,sz1,ret,H5S_ALL_F,H5S_ALL_F)
236         if(ret.lt.0)then
237           write(*,'(a,a)')'H5Dread error: '
238      +    ,'Pixel Data Quality'
239           call exit(1)
240         endif
241         call H5Dclose_f(dhnd,ret)
242         ! separate
243         do j=1,num
244           do i=1,AM2_DEF_SNUM_LO
245             pdq1(i,j)=pdqtmp(i,num*0+j)
246             pdq2(i,j)=pdqtmp(i,num*1+j)
247           enddo
248         enddo
249       endif
```

L2 品質フラグには、アルゴリズム開発者が設定した物理量算出に係る補足情報が格納されています。
0～15 は OK 状態、16～255 は NG 状態を現します。
NG 状態の場合、物理量には欠損値(-32768)または異常値(-32767)が格納されています。
L2 品質フラグの詳細については、「AMSR2 高次プロダクトフォーマット説明書」(*1)を参照ください。
(*1)http://suzaku.eorc.jaxa.jp/GCOM_W/data/data_w_format_j.html

◇終了処理　※左側の数字は行番号です

ファイルをクローズします。HDF5 を終了します。

```
1027 C close
1028       call H5Fclose_f(fhnd,ret)
1029       call H5close_f(ret)
```

◇ファイルのクローズ
call H5Fclose_f(fhnd,ret)
fhnd:ファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇HDF5 の終了
call H5close_f(ret)
ret: [戻り値]失敗の場合は負の値

８．３．２ コンパイル方法(build_readL2L_hdf5_f.sh 解説)

コンパイルに使用するスクリプト build_readL2L_hdf5_f.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～8 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 8 SZIP=/home/user1/util/szip_2.1
 9
```

11～12 行目に、使用するコンパイラを指定します。
時刻変換関数が C 言語のため、C コンパイラも指定します。
インテルコンパイラ(ifort/icc)または PG コンパイラ(pgf90/pgcc)を指定します。

```
10 # compiler
11 fc=ifort
12 cc=icc
13
14 # source filename
15 fsrc="readL2L_hdf5.f"
16 csrc="amsr2time_.c"
17 obj=`echo $csrc|sed "s/¥.c/.o/g"`
18
19 # output filename
20 out=readL2L_hdf5_f
21
22 # library order
23 lib="-lhdf5hl_fortran -lhdf5_hl -lhdf5_fortran -lhdf5 -lsz -lz -lm"
24
25 # c compile
26 cmd="$cc -g -c $csrc"
27 echo $cmd
28 $cmd
29
30 # f compile
31 cmd="$fc -g $fsrc $obj -o $out -I$HDF5/include -I$SZIP/include -L$HDF5/lib
-L$SZIP/lib $lib"
32 echo $cmd
33 $cmd
34
35 # garbage
36 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL2L_hdf5_f.sh
icc -g -c amsr2time_.c
ifort -g readL2L_hdf5.f amsr2time_.o -o readL2L_hdf5_f
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lhdf5hl_fortran -lhdf5_hl -lhdf5_fortran -lhdf5 -lsz -lz -lm
```

８．３．３ プログラム実行結果のサンプル

サンプルプログラムでは固定配列を多数使用しているため、環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞

```
$ unlimit
```

＜sh/bash 環境の場合＞

※以下のコマンドを順番に、4 つとも実行してください。

```
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited
```

```
$ ./readL2L_hdf5_f GW1AM2_201303011809_125D_L2SGCLWLA0000000.h5
input file: GW1AM2_201303011809_125D_L2SGCLWLA0000000.h5
GranuleID(RETRIEVE FROM FILENAME): GW1AM2_201303011809_125D_L2SGCLWLA0000000
NumberOfScans(RETRIEVE BY ARRAY SIZE):         1972
OverlapScans(FIXED VALUE):            0
limit of NumberOfScans =   2200
amsr2time: AMSR2_LEAP_DATA = /export/emc3/util/common/AMTK_AMSR2_DATA/leapsec.dat
amsr2time: year=1993 month= 7 tai93sec=   15638401.00
amsr2time: year=1994 month= 7 tai93sec=   47174402.00
amsr2time: year=1996 month= 1 tai93sec=   94608003.00
amsr2time: year=1997 month= 7 tai93sec=  141868804.00
amsr2time: year=1999 month= 1 tai93sec=  189302405.00
amsr2time: year=2006 month= 1 tai93sec=  410227206.00
amsr2time: year=2009 month= 1 tai93sec=  504921607.00
amsr2time: year=2012 month= 7 tai93sec=  615254408.00
amsr2time: number of leap second = 8
time(scan=1): 2013/03/01 18:09:10
latlon(pixel=1,scan=1): (  84.4574, -78.1076)
geo1(pixel=1,scan=1): -32767.000 [Kg/m2] (PDQ:112)
```

８．４ L2 高解像度データ読込み

降水量(PRC)が L2 高解像度です。
積算雲水量(CLW)・海氷密接度(SIC)・土壌水分量(SMC)・積雪深(SND)・海面水温(SST)・海上風速(SSW)・可降水量(TPW)は L2 低解像度を参照ください。

８．４．１ サンプルプログラム readL2H_hdf5.f 解説

サンプルプログラム readL2H_hdf5.f では、L2 高解像度データファイルから、以下の格納データを読込んで、内容をテキスト表示します。
TAI93 形式時刻の変換には、サブルーチンを用意しています。

メタデータ	格納データ
※HDF5 の FORTRAN90 ライブラリでは可変長文字列メタデータを読込めません。 ＊ NumberOfScans は配列サイズから調べます。 以下は固定値を使用します。 ＊ OverlapScans	＊ Scan Time ＊ Latitude of Observation Point for 89A － Latitude of Observation Point for 89B ＊ Longitude of Observation Point for 89A － Longitude of Observation Point for 89B ＊ Geophysical Data for 89A － Geophysical Data for 89B ＊ Pixel Data Quality for 89A － Pixel Data Quality for 89B P.15 基礎知識編３．１０参照

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の HDF5 関数は、最初に使用する際に使用説明を記載します。

HDF5 ライブラリでは、ファイルをオープンした後に、各データについてもオープンとクローズを繰り返します。各データの読込みは以下のような流れになります。

データをオープンする →必要な場合はスケールを取得する
→ データを読込む → データをクローズする

◇変数宣言　※左側の数字は行番号です

```
 1      program main
 2      use hdf5
 3      implicit none
 4 C include
 5      include 'amsr2time_f.h'
 :
 8 C fixed value
 9      integer(4),parameter::LMT=2200 ! limit of NumberOfScans
10      integer(4),parameter::AM2_DEF_SNUM_HI=486 ! high resolution pixel width
11      integer(4),parameter::AM2_DEF_SNUM_LO=243 ! low  resolution pixel width
 :
14 C interface variable
15      integer(4) i,j          ! loop variable
16      integer(4) ret          ! return status
17      character(len=512) buf  ! text buffer
18      character(len=512) fn   ! filename
19      integer(HID_T) fhnd     ! file handle
20      integer(HID_T) ahnd     ! attribute handle
21      integer(HID_T) atyp     ! attribute type
22      integer(HID_T) dhnd     ! dataset handle
23      integer(HID_T) shnd     ! dataspace handle
24      integer(HSIZE_T) sz1(3) ! array size 1
25      integer(HSIZE_T) sz2(3) ! array size 2
 :
35      integer(4) num          ! NumberOfScans
36      integer(4) ovr          ! OverlapScans
37      parameter(ovr=0)
 :
41 C array data
42      type(AM2_COMMON_SCANTIME) st(LMT) ! scantime
43      real(4) lat89a(AM2_DEF_SNUM_HI,LMT)  ! lat for 89a
44      real(4) lat89b(AM2_DEF_SNUM_HI,LMT)  ! lat for 89b
45      real(4) lon89a(AM2_DEF_SNUM_HI,LMT)  ! lon for 89a
46      real(4) lon89b(AM2_DEF_SNUM_HI,LMT)  ! lon for 89b
47      real(4) geo1_89a(AM2_DEF_SNUM_HI,LMT)
48      real(4) geo1_89b(AM2_DEF_SNUM_HI,LMT)
49      integer(4) pdq1_89a(AM2_DEF_SNUM_HI,LMT)
50      integer(4) pdq1_89b(AM2_DEF_SNUM_HI,LMT)
```

HDF5 用 use 文を宣言します。

時刻変換用サブルーチンヘッダーを
インクルードします。

標準プロダクトのスキャン数上限は
2200 で充分ですが、
準リアルプロダクトを使用する場合は、
2 周回程度繋がる場合があるので、
LMT=9000 としてください。
(標準の 4 パス分程度)

HDF5 用インターフェイス
変数を宣言します。

スキャン数
オーバーラップスキャン数

メタデータ用変数を宣言します。
※HDF5 の FORTRAN90 ライブラリでは
可変長文字列のメタデータを読込めな
いので、後で固定値を設定します。

時刻データには AM2_COMMON_SCANTIME 構造体を使います。
配列の次元数はデータによって異なります。
スキャン数はパスによって若干上下するので、上限を決めて宣言しておきます。
ここでは上限として LMT=2200 を設定しています。
AM2_DEF_SNUM_HI は高解像度ピクセル数(486)です。
AM2_DEF_SNUM_LO は低解像度ピクセル数(243)です。

時刻
緯度
経度
物理量
品質フラグ

配列データ用変数
を宣言します。

◇開始処理　※左側の数字は行番号です

HDF5 の初期化処理を行います。

◇HDF5 の初期化
call H5open_f(ret)
ret: [戻り値]失敗の場合は負の値

```
59 C hdf5 initialize
60       call H5open_f(ret)
61       if(ret.lt.0)then
62         write(*,'(a,i12)')'h5open_f error: ',ret
63         call exit(1)
64       endif
```

ファイルをオープンします。

◇HDF5 ファイルオープン
call H5Fopen_f(fn,label1,fhnd,ret,label2)
fn: オープンするファイル名を指定します
label1: アクセスモードを指定します H5F_ACC_RDONLY_F で読込専用になります
fhnd: [戻り値]開いたファイルのハンドル値
ret: [戻り値]失敗の場合は負の値
label2: H5P_DEFAULT_F を指定します

```
65 C open
66       call H5Fopen_f(fn,H5F_ACC_RDONLY_F,fhnd,ret,H5P_DEFAULT_F)
67       if(ret.lt.0)then
68         write(*,'(a,a)')'H5Fopen error: ',fn(1:len_trim(fn))
69         call exit(1)
70       endif
```

◇NumberOfScans の設定　※左側の数字は行番号です

HDF5 の FORTRAN90 ライブラリでは可変長文字列のメタデータを読み込めないので、
NumberOfScans を配列サイズから調べます。ここでは"Scan Time"のサイズを調べています。
配列サイズを取得するには、データセットをオープンして、データスペースをオープンしてから、
データセットのサイズ取得関数を使用します。不要になったハンドルはクローズします。

◇データセットのオープン
call H5Dopen_f(fhnd, nam, dhnd, ret)
fhnd: ファイルハンドル値を指定します
nam: データセット名を指定します
dhnd: [戻り値]データセットのハンドル値
ret: [戻り値]失敗の場合は負の値

◇データスペースのオープン
call H5Dget_space_f(dhnd, shnd, ret)
dhnd: データセットハンドル値を指定します
shnd: [戻り値]データスペースのハンドル値
ret: [戻り値]失敗の場合は負の値

◇データセットのサイズ取得
call H5Sget_simple_extent_dims_f(shnd, sz1, sz2, ret)
shnd: データスペースハンドル値を指定します
sz1: [戻り値]データセットの各次元サイズ
sz2: [戻り値]データセットの各次元の最大サイズ
ret: [戻り値]失敗の場合は負の値

```
82 C HDF5 FORTRAN LIBRARY CAN'T RETRIEVE VARIABLE STRING !
83 C calculate NumberOfScans from array size and OverlapScans
84       call H5Dopen_f(fhnd,'Scan Time',dhnd,ret)
85       call H5Dget_space_f(dhnd,shnd,ret)
86       call H5Sget_simple_extent_dims_f(shnd,sz1,sz2,ret)
87       if(ret.lt.0)then
88         write(*,'(a)')'H5Sget_simple_extent_dims error: Scan Time'
89         call exit(1)
90       endif
91       call H5Sclose_f(shnd,ret)
92       call H5Dclose_f(dhnd,ret)
93       num=sz1(1)-ovr*2
94       write(*,'(a,i12)')'NumberOfScans(RETRIEVE BY ARRAY SIZE): ',num
95       write(*,'(a,i12)')'OverlapScans(FIXED VALUE): ',ovr
```

NumberOfScans は、
「配列のスキャン数－(OverlapScans×２)」となります。

◇データスペースのクローズ
call H5Sclose_f(shnd, ret)
shnd: データスペースハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇データセットのクローズ
call H5Dclose_f(dhnd, ret)
dhnd: データセットハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇時刻データ読込み　※左側の数字は行番号です

データを読込むためには、データセットを H5Dopen()でオープンして、読込み用変数の各次元サイズを配列変数に設定しておきます。
時刻データの格納形式は TAI93 形式なので年/月/日/時/分/秒に変換します。

◇データ読込み
call H5Dread_f(dhnd,dtyp,buf,sz1,ret,label1,label2)
dhnd: データセットハンドル値を指定します
dtyp: データタイプ値を指定します
buf: 読込み用の変数を指定します
sz1: buf の各次元サイズを設定しておきます
ret: [戻り値]失敗の場合は負の値
label1,label2: 部分配列を読込む場合に使用します
全て読込む場合は、どちらも H5S_ALL_F を指定します

```
103 C read array: scantime
104       ! read
105       call H5Dopen_f(fhnd,'Scan Time',dhnd,ret)
106       sz1(1)=LMT
          call H5Dread_f(dhnd
         +,H5T_NATIVE_DOUBLE,r8d1,sz1,ret,H5S_ALL_F,H5S_ALL_F)
109       if(ret.lt.0)then
110         write(*,'(a,a)')'H5Dread error: ','Scan Time'
111         call exit(1)
112       endif
113       call H5Dclose_f(dhnd,ret)
114       ! convert
          call amsr2time(num,r8d1,st)
          ! sample display
117       write(*,'(a,i4.4,"/",i2.2,"/",i2.2," ",i2.2,":",i2.2,":",i2.2)')
118      +'time(scan=1): '
119      +,st(1)%year
120      +,st(1)%month
121      +,st(1)%day
122      +,st(1)%hour
123      +,st(1)%minute
124      +,st(1)%second
```

読込み

時刻変換

AMSR2 プロダクトの時刻格納形式は、TAI93 形式と呼ばれる、うるう秒を含めた 1993/01/01 からの通算秒です。
このままでは日時情報として扱い難いので、
このサンプルプログラムでは年/月/日/時/分/秒に変換するサブルーチンを用意しています。この時刻変換は AMTK を使用する場合は自動的に行われます。
→ P.11 基礎知識編３．６参照

◆時刻形式の変換
call amsr2time(num,in,out)
num: スキャン数を指定します
in: TAI93 形式の時刻データを指定します
out: [戻り値]AM2_COMMON_SCANTIME 構造体で年/月/日/時/分/秒データが返されます

◇緯度経度データ読込み　※左側の数字は行番号です

緯度経度データを読込みます。 → P.15 基礎知識編３．１０参照

緯度読込み

経度読込み

```
125 C read array: latlon for 89a
126       ! read lat
127       call H5Dopen_f(fhnd
128      +,'Latitude of Observation Point for 89A',dhnd,ret)
129       sz1(1)=LMT
130       sz1(2)=AM2_DEF_SNUM_HI
131       call H5Dread_f(dhnd
132      +,H5T_NATIVE_REAL,lat89a,sz1,ret,H5S_ALL_F,H5S_ALL_F)
133       if(ret.lt.0)then
134         write(*,'(a,a)')'H5Dread error: '
135      +  ,'Latitude of Observation Point for 89A'
136         call exit(1)
137       endif
138       call H5Dclose_f(dhnd,ret)
139       ! read lon
140       call H5Dopen_f(fhnd
141      +,'Longitude of Observation Point for 89A',dhnd,ret)
142       sz1(1)=LMT
143       sz1(2)=AM2_DEF_SNUM_HI
144       call H5Dread_f(dhnd
145      +,H5T_NATIVE_REAL,lon89a,sz1,ret,H5S_ALL_F,H5S_ALL_F)
146       if(ret.lt.0)then
147         write(*,'(a,a)')'H5Dread error: '
148      +  ,'Longitude of Observation Point for 89A'
149         call exit(1)
150       endif
151       call H5Dclose_f(dhnd,ret)
152       ! sample display
153       write(*,'(a,"(",f9.4,",",f9.4,")")')'latlon89a(pixel=1,scan=1): '
154      +,lat89a(1,1),lon89a(1,1)
```

◇物理量データ読込み　※左側の数字は行番号です

物理量データを読込みます。

物理量は格納型が 2byte 符号あり整数(-32768～32767)で、スケール値が設定されているので、
物理量に付属しているアトリビュートにアクセスして、スケール値を読み出します。
スケールを適用する際は、欠損値(-32768)と異常値(-32767)は除外します。

◇アトリビュートのオープン
call H5Aopen_f(dhnd, nam, ahnd, ret)
dhnd: データセットハンドル値を指定します
nam: アトリビュート名を指定します
ahnd: [戻り値]アトリビュートのハンドル値
ret: [戻り値]失敗の場合は負の値

◇アトリビュートのクローズ
call H5Aclose_f(ahnd, ret)
ahnd:アトリビュートハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇アトリビュート読込み
call H5Aread_f(ahnd, atyp, buf, sz1, ret)
ahnd: アトリビュートハンドル値を指定します
otyp: 出力変数の型を指定します
buf: 出力変数を指定します
sz1: buf の各次元サイズを設定しておきます(スカラ読込みでは無視されます)
ret: [戻り値]失敗の場合は負の値

```
185 C read array: geophysical data for 1 layer for 89a
186       call H5Dopen_f(fhnd,'Geophysical Data for 89A',dhnd,ret)
187       ! get scale
188       call H5Aopen_f(dhnd,'SCALE FACTOR',ahnd,ret)
189       call H5Aread_f(ahnd,H5T_NATIVE_REAL,sca,sz1,ret)
190       call H5Aclose_f(ahnd,ret)
191       ! read
192       sz1(1)=LMT
193       sz1(2)=AM2_DEF_SNUM_HI
          call H5Dread_f(dhnd
         +,H5T_NATIVE_REAL,geo1_89a,sz1,ret,H5S_ALL_F,H5S_ALL_F)
196       if(ret.lt.0)then
197         write(*,'(a,a)')'H5Dread error: '
198      +  ,'Geophysical Data for 89A'
199         call exit(1)
200       endif
201       call H5Dclose_f(dhnd,ret)
202       ! change scale
203       do j=1,num
204         do i=1,AM2_DEF_SNUM_HI
            if(geo1_89a(i,j).gt.-32767)geo1_89a(i,j)=geo1_89a(i,j)*sca
          enddo
207       enddo
```

スケール読込み

読込み

スケール処理

このスケール処理は AMTK を使用する場合は
自動的に行われます。
→ P.16 基礎知識編３． １２参照

欠損値(-32768)と異常値(-32767)は除外して、
スケールを適用します。

◇L2 品質フラグデータ読込み　※左側の数字は行番号です

L2 品質フラグデータを読込みます。

```
231 C read array: pixel data quality for 1 layer for 89a
232       call H5Dopen_f(fhnd,'Pixel Data Quality for 89A',dhnd,ret)
233       sz1(1)=LMT
234       sz1(2)=AM2_DEF_SNUM_HI
235       call H5Dread_f(dhnd
236      +,H5T_NATIVE_INTEGER,pdq1_89a,sz1,ret,H5S_ALL_F,H5S_ALL_F)
237       if(ret.lt.0)then
238         write(*,'(a,a)')'H5Dread error: '
239      +  ,'Pixel Data Quality for 89A'
240         call exit(1)
241       endif
242       call H5Dclose_f(dhnd,ret)
```

L2 品質フラグには、アルゴリズム開発者が設定した物理量算出に係る補足情報が格納されています。
0～15 は OK 状態、16～255 は NG 状態を現します。
NG 状態の場合、物理量には欠損値(-32768)または異常値(-32767)が格納されています。
L2 品質フラグの詳細については、「AMSR2 高次プロダクトフォーマット説明書」(*1)を参照ください。
(*1)http://suzaku.eorc.jaxa.jp/GCOM_W/data/data_w_format_j.html

◇終了処理　※左側の数字は行番号です

ファイルをクローズします。HDF5 を終了します。

```
261 C close
262       call H5Fclose_f(fhnd,ret)
263       call H5close_f(ret)
```

◇ファイルのクローズ
call H5Fclose_f(fhnd,ret)
fhnd:ファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇HDF5 の終了
call H5close_f(ret)
ret: [戻り値]失敗の場合は負の値

８．４．２ コンパイル方法(build_readL2H_hdf5_f.sh 解説)

コンパイルに使用するスクリプト build_readL2H_hdf5_f.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～8 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 8 SZIP=/home/user1/util/szip_2.1
 9
```

11～12 行目に、使用するコンパイラを指定します。
時刻変換関数が C 言語のため、C コンパイラも指定します。
インテルコンパイラ(ifort/icc)または PG コンパイラ(pgf90/pgcc)を指定します。

```
10 # compiler
11 fc=ifort
12 cc=icc
13
14 # source filename
15 fsrc="readL2H_hdf5.f"
16 csrc="amsr2time_.c"
17 obj=`echo $csrc|sed "s/¥.c/.o/g"`
18
19 # output filename
20 out=readL2H_hdf5_f
21
22 # library order
23 lib="-lhdf5hl_fortran -lhdf5_hl -lhdf5_fortran -lhdf5 -lsz -lz -lm"
24
25 # c compile
26 cmd="$cc -g -c $csrc"
27 echo $cmd
28 $cmd
29
30 # f compile
31 cmd="$fc -g $fsrc $obj -o $out -I$HDF5/include -I$SZIP/include -L$HDF5/lib
-L$SZIP/lib $lib"
32 echo $cmd
33 $cmd
34
35 # garbage
36 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL2H_hdf5_f.sh
icc -g -c amsr2time_.c
ifort -g readL2H_hdf5.f amsr2time_.o -o readL2H_hdf5_f
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lhdf5hl_fortran -lhdf5_hl -lhdf5_fortran -lhdf5 -lsz -lz -lm
```

８．４．３ プログラム実行結果のサンプル

サンプルプログラムでは固定配列を多数使用しているため、環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞

```
$ unlimit
```

＜sh/bash 環境の場合＞

※以下のコマンドを順番に、4 つとも実行してください。

```
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited
```

```
$ ./readL2H_hdf5_f GW1AM2_201303011809_125D_L2SGPRCHA0000000.h5
input file: GW1AM2_201303011809_125D_L2SGPRCHA0000000.h5
GranuleID(RETRIEVE FROM FILENAME): GW1AM2_201303011809_125D_L2SGPRCHA0000000
NumberOfScans(RETRIEVE BY ARRAY SIZE):         1972
OverlapScans(FIXED VALUE):            0
limit of NumberOfScans =   2200
amsr2time: AMSR2_LEAP_DATA = /export/emc3/util/common/AMTK_AMSR2_DATA/leapsec.dat
amsr2time: year=1993 month= 7 tai93sec=   15638401.00
amsr2time: year=1994 month= 7 tai93sec=   47174402.00
amsr2time: year=1996 month= 1 tai93sec=   94608003.00
amsr2time: year=1997 month= 7 tai93sec=  141868804.00
amsr2time: year=1999 month= 1 tai93sec=  189302405.00
amsr2time: year=2006 month= 1 tai93sec=  410227206.00
amsr2time: year=2009 month= 1 tai93sec=  504921607.00
amsr2time: year=2012 month= 7 tai93sec=  615254408.00
amsr2time: number of leap second = 8
time(scan=1): 2013/03/01 18:09:10
latlon89a(pixel=1,scan=1): (  84.4188, -77.9502)
latlon89b(pixel=1,scan=1): (  84.3305, -78.8925)
geo1_89a(pixel=1,scan=1): -32767.0 [mm/h] (PDQ: 16)
geo1_89b(pixel=1,scan=1): -32767.0 [mm/h] (PDQ: 16)
```

８．５ L3 輝度温度データ読込み

８．５．１ サンプルプログラム readL3B_hdf5.f 解説

サンプルプログラム readL3B_hdf5.f では、L3 輝度温度データファイルから、以下の格納データを読込んで、内容をテキスト表示します。

メタデータ	格納データ
※HDF5 の FORTRAN90 ライブラリでは可変長文字列メタデータを読込めません。	* Brightness Temperature (H) - Brightness Temperature (V)

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の HDF5 関数は、最初に使用する際に使用説明を記載します。

HDF5 ライブラリでは、ファイルをオープンした後に、各データについてもオープンとクローズを繰り返します。各データの読込みは以下のような流れになります。

データをオープンする →必要な場合はスケールを取得する
→ データを読込む → データをクローズする

◇変数宣言　※左側の数字は行番号です

```
 1      program main
 2      use hdf5
 3      implicit none
 :
 9 C interface variable
10      integer(4) i,j          ! loop variable
11      integer(4) ret          ! return status
12      character(len=512) buf  ! text buffer
13      character(len=512) fn   ! filename
14      integer(HID_T) fhnd     ! file handle
15      integer(HID_T) ahnd     ! attribute handle
16      integer(HID_T) atyp     ! attribute type
17      integer(HID_T) dhnd     ! dataset handle
18      integer(HID_T) shnd     ! dataspace handle
19      integer(HSIZE_T) sz1(3) ! array size 1
20      integer(HSIZE_T) sz2(3) ! array size 2
21      integer(4) x            ! grid size x
22      integer(4) y            ! grid size y
 :
28 C meta data
29 C HDF5 FORTRAN LIBRARY CAN'T RETRIEVE VARIABLE STRING !
30 C retrieve GranuleID from filename
31 C     character(len=512) geo  ! GeophysicalName
32      character(len=512) gid  ! GranuleID
 :
35 C array data
36      real(4),allocatable::tbH(:,:) 輝度温度
37      real(4),allocatable::tbV(:,:)
```

HDF5 用 use 文を宣言します。

HDF5 用インターフェイス
変数を宣言します。

配列データ用変数
を宣言します。

この時点では、配列データ用のメモリ領域は確保されていません。
後程、配列サイズを調べてから、メモリ領域の確保を行います。

◇開始処理　※左側の数字は行番号です

HDF5 の初期化処理を行います。

◇HDF5 の初期化
call H5open_f(ret)
ret: [戻り値]失敗の場合は負の値

```
46 C hdf5 initialize
47      call H5open_f(ret)
48      if(ret.lt.0)then
49        write(*,'(a,i12)')'h5open_f error: ',ret
50        call exit(1)
51      endif
```

ファイルをオープンします。

◇HDF5 ファイルオープン
call H5Fopen_f(fn,label1,fhnd,ret,label2)
fn: オープンするファイル名を指定します
label1: アクセスモードを指定します H5F_ACC_RDONLY_F で読込専用になります
fhnd: [戻り値]開いたファイルのハンドル値
ret: [戻り値]失敗の場合は負の値
label2: H5P_DEFAULT_F を指定します

```
52 C open
53      call H5Fopen_f(fn,H5F_ACC_RDONLY_F,fhnd,ret,H5P_DEFAULT_F)
54      if(ret.lt.0)then
55        write(*,'(a,a)')'H5Fopen error: ',fn(1:len_trim(fn))
56        call exit(1)
57      endif
```

◇配列サイズ取得とメモリ領域確保　※左側の数字は行番号です

配列サイズを取得します。

◇データセットのオープン
call H5Dopen_f(fhnd,nam,dhnd,ret)
fhnd: ファイルハンドル値を指定します
nam: データセット名を指定します
dhnd: [戻り値]データセットのハンドル値
ret: [戻り値]失敗の場合は負の値

◇データスペースのオープン
call H5Dget_space_f(dhnd,shnd,ret)
dhnd: データセットハンドル値を指定します
shnd: [戻り値]データスペースのハンドル値
ret: [戻り値]失敗の場合は負の値

◇データセットのサイズ取得
call H5Sget_simple_extent_dims_f(shnd,sz1,sz2,ret)
shnd: データスペースハンドル値を指定します
sz1: [戻り値]データセットの各次元サイズ
sz2: [戻り値]データセットの各次元の最大サイズ
ret: [戻り値]失敗の場合は負の値

```
75 C get grid size
76       call H5Dopen_f(fhnd,'Brightness Temperature (H)',dhnd,ret)
77       call H5Dget_space_f(dhnd,shnd,ret)
78       call H5Sget_simple_extent_dims_f(shnd,sz1,sz2,ret)
79       if(ret.lt.0)then
80         write(*,'(a,a)')'H5Sget_simple_extent_dims error: '
81      +  ,'Brightness Temperature (H)'
82         call exit(1)
83       endif
84       call H5Sclose_f(shnd,ret)
85       call H5Dclose_f(dhnd,ret)
86       x=sz1(1);
87       y=sz1(2);
88       write(*,'(a,i12)')'grid size x: ',x
89       write(*,'(a,i12)')'grid size y: ',y
```

◇データスペースのクローズ
call H5Sclose_f(shnd,ret)
shnd: データスペースハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇データセットのクローズ
call H5Dclose_f(dhnd,ret)
dhnd: データセットハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

メモリ領域を確保します。

```
90 C memory allocate
91       allocate(tbH(x,y),stat=ret)
92       if(ret.ne.0)then
93         write(*,'(a)')'memory allocate error: tbH'
94         call exit(1)
95       endif
```

◇輝度温度データ読込み　※左側の数字は行番号です

輝度温度データを読込みます。

輝度温度は格納型が 2byte 符号なし整数(0～65535)で、スケール値が設定されているので、
輝度温度に付属しているアトリビュートにアクセスして、スケール値を読み出します。
スケールを適用する際は、欠損値(65535)と異常値(65534)は除外します。

◇アトリビュートのオープン
call H5Aopen_f(dhnd, nam, ahnd, ret)
dhnd: データセットハンドル値を指定します
nam: アトリビュート名を指定します
ahnd: [戻り値]アトリビュートのハンドル値
ret: [戻り値]失敗の場合は負の値

◇アトリビュートのクローズ
call H5Aclose_f(ahnd, ret)
ahnd:アトリビュートハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇アトリビュート読込み
call H5Aread_f(ahnd, atyp, buf, sz1, ret)
ahnd: アトリビュートハンドル値を指定します
otyp: 出力変数の型を指定します
buf: 出力変数を指定します
sz1: buf の各次元サイズを設定しておきます(スカラ読込みでは無視されます)
ret: [戻り値]失敗の場合は負の値

◇データ読込み
call H5Dread_f(dhnd, dtyp, buf, sz1, ret, label1, label2)
dhnd: データセットハンドル値を指定します
dtyp: データタイプ値を指定します
buf: 読込み用の変数を指定します
sz1: buf の各次元サイズを設定しておきます
ret: [戻り値]失敗の場合は負の値
label1, label2: 部分配列を読込む場合に使用します
全て読込む場合は、どちらも H5S_ALL_F を指定します

```
101 C read horizontal
102       call H5Dopen_f(fhnd,'Brightness Temperature (H)',dhnd,ret)
103       ! get scale
104       call H5Aopen_f(dhnd,'SCALE FACTOR',ahnd,ret)
105       call H5Aread_f(ahnd,H5T_NATIVE_REAL,sca,sz1,ret)
106       call H5Aclose_f(ahnd,ret)
107       ! read
108       sz1(1)=y
109       sz1(2)=x
          call H5Dread_f(dhnd
111      +,H5T_NATIVE_REAL,tbH,sz1,ret,H5S_ALL_F,H5S_ALL_F)
112       if(ret.lt.0)then
113         write(*,'(a,a)')'H5Dread error: '
114      +  ,'Brightness Temperature (H)'
115         call exit(1)
116       endif
117       call H5Dclose_f(dhnd,ret)
118       ! change scale
119       do j=1,y
            do i=1,x
              if(tbH(i,j).lt.65534)tbH(i,j)=tbH(i,j)*sca
122         enddo
123       enddo
```

スケール読込み

読込み

スケール処理

このスケール処理は AMTK を使用する場合は
自動的に行われます。
→ P.16 基礎知識編３． １２参照

欠損値(65535)と異常値(65534)は除外して、
スケールを適用します。

◇終了処理　※左側の数字は行番号です

メモリを開放します。

```
239 C memory free
240       deallocate(tbH)
241       deallocate(tbV)
```

ファイルをクローズします。HDF5 を終了します。

◇ファイルのクローズ
call H5Fclose_f(fhnd,ret)
fhnd:ファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇HDF5 の終了
call H5close_f(ret)
ret: [戻り値]失敗の場合は負の値

```
242 C close
243       call H5Fclose_f(fhnd,ret)
244       call H5close_f(ret)
245       end
```

８．５．２ コンパイル方法(build_readL3B_hdf5_f.sh 解説)

コンパイルに使用するスクリプト build_readL3B_hdf5_f.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～8 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 8 SZIP=/home/user1/util/szip_2.1
 9
```

11 行目に、使用するコンパイラを指定します。
インテルコンパイラ(ifort)または PG コンパイラ(pgf90)を指定します。

```
10 # compiler
11 fc=ifort
12
13 # source filename
14 fsrc="readL3B_hdf5.f"
15
16 # output filename
17 out=readL3B_hdf5_f
18
19 # library order
20 lib="-lhdf5hl_fortran -lhdf5_hl -lhdf5_fortran -lhdf5 -lsz -lz -lm"
21
22 # f compile
23 cmd="$fc -g $fsrc -o $out -I$HDF5/include -I$SZIP/include -L$HDF5/lib
-L$SZIP/lib $lib"
24 echo $cmd
25 $cmd
26
27 # garbage
28 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL3B_hdf5_f.sh
ifort -g readL3B_hdf5.f -o readL3B_hdf5_f
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lhdf5hl_fortran -lhdf5_hl -lhdf5_fortran -lhdf5 -lsz -lz -lm
```

８．５．３ プログラム実行結果のサンプル

環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞
$ unlimit

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、
4 つとも実行してください。
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited

```
$ ./readL3B_hdf5_f GW1AM2_20130200_01M_EQMA_L3SGT06LA1110110.h5
input file: GW1AM2_20130200_01M_EQMA_L3SGT06LA1110110.h5
GranuleID(RETRIEVE FROM FILENAME): GW1AM2_20130200_01M_EQMA_L3SGT06LA1110110
grid size x:         1440
grid size y:          720

ASCII ART OF HORIZONTAL BRIGHTNESS TEMPERATURE (X/ 20GRID Y/ 40GRID)
+------------------------------------------------------------------------+
|                                                                        |
|4112333444434444444444444444444444444444444444444444444444433443222222444444|
|1111343111134444444444444444444444444444444444444444444444444434233343111|
|1245234455555555444444444444444434312343444144555444444444442112111111111|
|1545444444444444444444444445441111111111111111113554444444442221111111112|
|4122425544144445444444453411311111111111111111111455544445511111111111115|
|4444444442444555445555441111111111111111111111155555351111111111111124|
|4444444424441115551555311111111111111111121111111155151141111111111444|
|5555555555511111211113111211111111111111111111111111141554111111111155|
|1155555551111111111151553111111111111111111111111111111455555411111111|
|1115555531111111111111114211113121111111111111111111111555555551111111|
|1115555215111111111111114555551111111111111111111111111145555521111111|
|1111555111111111111111135555555111111111111111111111111154552111111111|
|1111111111111111111111111111111131111111111111111111155111111111111|
|1111111111111111111111111111111111111111111111111111135111111111111|
|1111111111111111111111111111111111111111111111111111111111111111111|
|3433234443333424343343444444443343321111111111111111113444444411111133|
|3333333333333333333333333333333343333333222223333333443344333333333333333|
+------------------------------------------------------------------------+
```

```
ASCII ART OF VERTICAL BRIGHTNESS TEMPERATURE (X/ 20GRID Y/ 40GRID)
+------------------------------------------------------------------------+
|                                                                        |
|43334444444445545544444444444444444444445444444444444444444433333244545|
|33334543333455455444454444444444444444444454544444444444444455443443443333|
|33553445555555555555455544545555444334545543455555444455444433333333333|
|35555555555555555545555555555433333333333333333345555555555433333333333333|
|43334355553555555555555545334333333333333333333355555555553333333333333335|
|55555555554555555555555553333333333333333333333355555453333333333333335|
|55555555355533355535554333333333333333333333333333553533433333333333555|
|65556665555333333333433343333333333333333333333333333535555333333333355|
|33555555633333333335355433333333333333333333333333333355555554333333333|
|33355555433333333333333433333433333333333333333333333355555556333333333|
|33355553353333333333333566665333333333333333333333333355555543333333333|
|33336553333333333333346666665333333333333333333333333555554333333333333|
|33333333333333333333333333333333343333333333333333333355333333333333333|
|33333333333333333333333333333333333333333333333333333345333333333333333|
|33333333333333333333333333333333333333333333333333333333333333333333333|
|44433444444444344444444444444444444433333333333333333334445555433333344|
|44444443344444444443333443444444444444444433444444444444444444444444444|
+------------------------------------------------------------------------+

[#]:missing
[ ]:out of observation
[1]: 50-100K
[2]:100-150K
[3]:150-200K
[4]:200-250K
[5]:250-300K
[6]:300-350K
[*]:other
```

８．６ L3 物理量データ読込み

８．６．１ サンプルプログラム readL3G_hdf5.f 解説

サンプルプログラム readL3G_hdf5.f では、L3 物理量データファイルから、以下の格納データを読込んで、内容をテキスト表示します。

メタデータ	格納データ
※HDF5 の FORTRAN90 ライブラリでは可変長文字列メタデータを読込めません。	* Geophysical Data

以下の説明では、プログラムの似たような繰返しを省略して、上記の*印を付けたデータ読込みについて抜粋して解説します。各々の HDF5 関数は、最初に使用する際に使用説明を記載します。

HDF5 ライブラリでは、ファイルをオープンした後に、各データについてもオープンとクローズを繰り返します。各データの読込みは以下のような流れになります。

データをオープンする →必要な場合はスケールを取得する
→ データを読込む → データをクローズする

◇変数宣言　※左側の数字は行番号です

```
 1      program main
 2      use hdf5
 3      implicit none
 :
 9 C interface variable
10      integer(4) i,j          ! loop variable
11      integer(4) ret          ! return status
12      character(len=512) buf  ! text buffer
13      character(len=512) fn   ! filename
14      integer(HID_T) fhnd     ! file handle
15      integer(HID_T) ahnd     ! attribute handle
16      integer(HID_T) atyp     ! attribute type
17      integer(HID_T) dhnd     ! dataset handle
18      integer(HID_T) shnd     ! dataspace handle
19      integer(HSIZE_T) sz1(3) ! array size 1
20      integer(HSIZE_T) sz2(3) ! array size 2
21      integer(4) x            ! grid size x
22      integer(4) y            ! grid size y
 :
30 C meta data
31 C HDF5 FORTRAN LIBRARY CAN'T RETRIEVE VARIABLE STRING !
32 C retrieve GranuleID from filename
33 C    character(len=512) geo  ! GeophysicalName
34      character(len=512) gid  ! GranuleID
 :
37 C array data
38      real(4),allocatable::geo1(:,:)
39      real(4),allocatable::geo2(:,:)
40      real(4),allocatable::geotmp(:,:,:)
```

HDF5 用 use 文を宣言します。

HDF5 用インターフェイス
変数を宣言します。

物理量

配列データ用変数
を宣言します。

この時点では、配列データ用のメモリ領域は確保されていません。
後程、配列サイズを調べてから、メモリ領域の確保を行います。

◇開始処理　※左側の数字は行番号です

HDF5 の初期化処理を行います。

◇HDF5 の初期化
call H5open_f(ret)
ret: [戻り値]失敗の場合は負の値

```
49 C hdf5 initialize
50      call H5open_f(ret)
51      if(ret.lt.0)then
52        write(*,'(a,i12)')'h5open_f error: ',ret
53        call exit(1)
54      endif
```

ファイルをオープンします。

◇HDF5 ファイルオープン
call H5Fopen_f(fn,label1,fhnd,ret,label2)
fn: オープンするファイル名を指定します
label1: アクセスモードを指定します H5F_ACC_RDONLY_F で読込専用になります
fhnd: [戻り値]開いたファイルのハンドル値
ret: [戻り値]失敗の場合は負の値
label2: H5P_DEFAULT_F を指定します

```
55 C open
56      call H5Fopen_f(fn,H5F_ACC_RDONLY_F,fhnd,ret,H5P_DEFAULT_F)
57      if(ret.lt.0)then
58        write(*,'(a,a)')'H5Fopen error: ',fn(1:len_trim(fn))
59        call exit(1)
60      endif
```

◇配列サイズ取得とメモリ領域確保　※左側の数字は行番号です

配列サイズを取得します。

◇データセットのオープン
call H5Dopen_f(fhnd,nam,dhnd,ret)
fhnd: ファイルハンドル値を指定します
nam: データセット名を指定します
dhnd: [戻り値]データセットのハンドル値
ret: [戻り値]失敗の場合は負の値

◇データスペースのオープン
call H5Dget_space_f(dhnd,shnd,ret)
dhnd: データセットハンドル値を指定します
shnd: [戻り値]データスペースのハンドル値
ret: [戻り値]失敗の場合は負の値

◇データセットのサイズ取得
call H5Sget_simple_extent_dims_f(shnd,sz1,sz2,ret)
shnd: データスペースハンドル値を指定します
sz1: [戻り値]データセットの各次元サイズ
sz2: [戻り値]データセットの各次元の最大サイズ
ret: [戻り値]失敗の場合は負の値

```
79 C get grid size
 80      call H5Dopen_f(fhnd,'Geophysical Data',dhnd,ret)
 81      call H5Dget_space_f(dhnd,shnd,ret)
 82      call H5Sget_simple_extent_dims_f(shnd,sz1,sz2,ret)
 83      if(ret.lt.0)then
 84        write(*,'(a,a)')'H5Sget_simple_extent_dims error: '
 85     +  ,'Geophysical Data'
 86        call exit(1)
 87      endif
 88      call H5Sclose_f(shnd,ret)
 89      call H5Dclose_f(dhnd,ret)
 90      x=sz1(2);
 91      y=sz1(3);
 92      write(*,'(a,i12)')'grid size x: ',x
 93      write(*,'(a,i12)')'grid size y: ',y
```

◇データスペースのクローズ
call H5Sclose_f(shnd,ret)
shnd: データスペースハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇データセットのクローズ
call H5Dclose_f(dhnd,ret)
dhnd: データセットハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

メモリ領域を確保します。

```
 94 C memory allocate layer 1
 95      allocate(geo1(x,y),stat=ret)
 96      if(ret.ne.0)then
 97        write(*,'(a)')'memory allocate error: geo1'
 98        call exit(1)
 99      endif
```

◇物理量データ読込み　※左側の数字は行番号です

物理量データを読込みます。
積雪深(SND)・海面水温(SST)は物理量が 2 層あるので、この 2 つとそれ以外で処理を分けています。
積雪深の 2 層目は積雪水量[cm]です。海面水温の 2 層目は 10GHz による SST[℃]です。

物理量は格納型が 2byte 符号あり整数(-32768～32767)で、スケール値が設定されているので、
輝度温度に付属しているアトリビュートにアクセスして、スケール値を読み出します。
スケールを適用する際は、欠損値(-32768)と異常値(-32767)は除外します。

◇アトリビュートのオープン
call H5Aopen_f(dhnd,nam,ahnd,ret)
dhnd: データセットハンドル値を指定します
nam: アトリビュート名を指定します
ahnd: [戻り値]アトリビュートのハンドル値
ret: [戻り値]失敗の場合は負の値

◇アトリビュートのクローズ
call H5Aclose_f(ahnd,ret)
ahnd:アトリビュートハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇アトリビュート読込み
call H5Aread_f(ahnd,atyp,buf,sz1,ret)
ahnd: アトリビュートハンドル値を指定します
otyp: 出力変数の型を指定します
buf: 出力変数を指定します
sz1: buf の各次元サイズを設定しておきます(スカラ読込みでは無視されます)
ret: [戻り値]失敗の場合は負の値

◇データ読込み
call H5Dread_f(dhnd,dtyp,buf,sz1,ret,label1,label2)
dhnd: データセットハンドル値を指定します
dtyp: データタイプ値を指定します
buf: 読込み用の変数を指定します
sz1: buf の各次元サイズを設定しておきます
ret: [戻り値]失敗の場合は負の値
label1,label2: 部分配列を読込む場合に使用します
全て読込む場合は、どちらも H5S_ALL_F を指定します

```
113 C read layer 1
114       if(gid(30:32).ne.'SND')then
115         call H5Dopen_f(fhnd,'Geophysical Data',dhnd,ret)
116         ! get scale
117         call H5Aopen_f(dhnd,'SCALE FACTOR',ahnd,ret)
118         call H5Aread_f(ahnd,H5T_NATIVE_REAL,sca,sz1,ret)
119         call H5Aclose_f(ahnd,ret)
120         ! read
121         sz1(1)=y
122         sz1(2)=x
123         call H5Dread_f(dhnd
124      +  ,H5T_NATIVE_REAL,geo1,sz1,ret,H5S_ALL_F,H5S_ALL_F)
125         if(ret.lt.0)then
126           write(*,'(a,a)')'H5Dread error: '
127      +    ,'Geophysical Data'
128           call exit(1)
129         endif
130         call H5Dclose_f(dhnd,ret)
131         ! change scale
132         do j=1,y
133           do i=1,x
134             if(geo1(i,j).gt.-32767)geo1(i,j)=geo1(i,j)*sca
135           enddo
136         enddo
137       endif
```

スケール読込み

読込み

スケール処理

このスケール処理は AMTK を使用する場合は
自動的に行われます。
→ P.16 基礎知識編３． １２参照

欠損値(-32768)と異常値(-32767)は除外して、
スケールを適用します。

2 層ある場合は、一時変数で全体を読込んだ後に、層ごとに分割しています。

```
138 C read layer 1 & 2
139      if(gid(30:32).eq.'SND')then
140        call H5Dopen_f(fhnd,'Geophysical Data',dhnd,ret)
141        ! get scale
142        call H5Aopen_f(dhnd,'SCALE FACTOR',ahnd,ret)
143        call H5Aread_f(ahnd,H5T_NATIVE_REAL,sca,sz1,ret)
144        call H5Aclose_f(ahnd,ret)
145        ! read
146        sz1(1)=y
147        sz1(2)=x
148        sz1(3)=2
149        call H5Dread_f(dhnd
150     +  ,H5T_NATIVE_REAL,geotmp,sz1,ret,H5S_ALL_F,H5S_ALL_F)
151        if(ret.lt.0)then
152          write(*,'(a,a)')'H5Dread error: '
153     +    ,'Geophysical Data'
154          call exit(1)
155        endif
156        call H5Dclose_f(dhnd,ret)
157        ! separate
158        do j=1,y
159          do i=1,x
160            geo1(i,j)=geotmp(1,i,j)
161            geo2(i,j)=geotmp(2,i,j)
162          enddo
163        enddo
164        ! change scale
165        do j=1,y
166          do i=1,x
167            if(geo1(i,j).gt.-32767)geo1(i,j)=geo1(i,j)*sca
168            if(geo2(i,j).gt.-32767)geo2(i,j)=geo2(i,j)*sca
169          enddo
170        enddo
171      endif
```

◇終了処理　※左側の数字は行番号です

メモリを開放します。

```
357 C memory free
358      deallocate(geo1)
359      if(gid(30:32).eq.'SND')then
360        deallocate(geo2)
361      endif
```

ファイルをクローズします。HDF5 を終了します。

◇ファイルのクローズ
call H5Fclose_f(fhnd,ret)
fhnd:ファイルハンドル値を指定します
ret: [戻り値]失敗の場合は負の値

◇HDF5 の終了
call H5close_f(ret)
ret: [戻り値]失敗の場合は負の値

```
362 C close
363      call H5Fclose_f(fhnd,ret)
364      call H5close_f(ret)
```

８．６．２ コンパイル方法(build_readL3G_hdf5_f.sh 解説)

コンパイルに使用するスクリプト build_readL3G_hdf5_f.sh の内容について以下に説明します。

※左側の数字は行番号です

```
 1 #!/bin/sh
 2
 3 ### environment
 4 export LANG=C
 5
```

7～8 行目に、インストールしたライブラリの場所を指定します。
指定したライブラリディレクトリ直下には include ディレクトリと lib ディレクトリが必要です。

```
 6 # library directory
 7 HDF5=/home/user1/util/hdf5_1.8.4-patch1
 8 SZIP=/home/user1/util/szip_2.1
 9
```

11 行目に、使用するコンパイラを指定します。
インテルコンパイラ(ifort)または PG コンパイラ(pgf90)を指定します。

```
10 # compiler
11 fc=ifort
12
13 # source filename
14 fsrc="readL3G_hdf5.f"
15
16 # output filename
17 out=readL3G_hdf5_f
18
19 # library order
20 lib="-lhdf5hl_fortran -lhdf5_hl -lhdf5_fortran -lhdf5 -lsz -lz -lm"
21
22 # f compile
23 cmd="$fc -g $fsrc -o $out -I$HDF5/include -I$SZIP/include -L$HDF5/lib
-L$SZIP/lib $lib"
24 echo $cmd
25 $cmd
26
27 # garbage
28 rm -f *.o
```

◇コンパイル実行サンプル　※読みやすいように改行を追加しています。

```
$ ./build_readL3G_hdf5_f.sh
ifort -g readL3G_hdf5.f -o readL3G_hdf5_f
-I/home/user1/util/hdf5_1.8.4-patch1/include
-I/home/user1/util/szip_2.1/include
-L/home/user1/util/hdf5_1.8.4-patch1/lib
-L/home/user1/util/szip_2.1/lib
-lhdf5hl_fortran -lhdf5_hl -lhdf5_fortran -lhdf5 -lsz -lz -lm
```

８．６．３ プログラム実行結果のサンプル

環境によっては、リソース不足でセグメントエラーが発生します。その場合は、以下の制限解除コマンドを実行してください。

＜csh/tcsh 環境の場合＞
```
$ unlimit
```

＜sh/bash 環境の場合＞
※以下のコマンドを順番に、4 つとも実行してください。
```
$ ulimit -d unlimited
$ ulimit -m unlimited
$ ulimit -s unlimited
$ ulimit -v unlimited
```

```
$ ./readL3G_hdf5_f GW1AM2_20130200_01M_EQMA_L3SGCLWLA0000000.h5
input file: GW1AM2_20130200_01M_EQMA_L3SGCLWLA0000000.h5
GranuleID(RETRIEVE FROM FILENAME): GW1AM2_20130200_01M_EQMA_L3SGCLWLA0000000
grid size x:         1440
grid size y:          720

ASCII ART OF GEOPHYSICAL DATA LAYER #1 (X/ 20GRID Y/ 40GRID)
+------------------------------------------------------------------------+
|                                                                        |
|#21#####################################################################|
|2212###1113#############################################################222|
|1#################################11######3################011#42223322|
|2##############################00101112413333233##############343413131#|
|#20#######2###############00#12234523442132111120#########3355421131121#|
|#########################4#32222232321001101111107#######14*41111201111##|
|########0###000###0####1111111121012011 0*131121100##1#11#01211110100###|
|###########1011111111#211#11111210011121333223224321000#1###1111000000##|
|11#######00010122125#3###1#12232321100010020110001011122#######222321111|
|000######122131112311112#01112#113122123231101000000000110########1001110|
|100####32#12111141111000######223232242121244133111001101 0######11111100|
|1000###121123221125110 0########131112232110121313122011010#####231111001|
|11131212231110111111211111110012102#12312212233112131110 21##011124223242|
|*33222511232222222223233331421022422322211024222222122113##0111421221132|
|10112211222111122211111111112223122322323312433212123232212233123232221 11|
|#################################3111221222211121122 1#######21221##|
|#####################################################################|
+------------------------------------------------------------------------+

[#]:missing
[ ]:out of observation
[0]:  0.000 -  0.030 Kg/m2
[1]:  0.030 -  0.060 Kg/m2
[2]:  0.060 -  0.090 Kg/m2
[3]:  0.090 -  0.120 Kg/m2
[4]:  0.120 -  0.150 Kg/m2
[5]:  0.150 -  0.180 Kg/m2
[*]:other
```

９．AMSR2 プロダクトデータ一覧

以下に、サンプルプログラムで取り扱ったデータ一覧を示します。AMSR2 プロダクトには、これ以外にもデータがあります。詳しくは、関連文書の AMTK 取扱い説明書等をご覧ください。関連文書は、GCOM-W1 データ提供サービス(http://gcom-w1.jaxa.jp/)からダウンロードできます。

表 10　サンプルプログラムで取り扱った L1B データ一覧

HDF5 データセット名 *は算出値のため非格納	HDF5 格納型	AMTK 出力型	AMTK アクセス関数	AMTK アクセスラベル	サイズ
Scan Time	Ｒ８	ＴＭ	AMTK_getScanTime()		SCAN
Latitude of Observation Point for 89A	Ｒ４	ＬＬ	AMTK_getLatLon()	AM2_LATLON_89A	486 x SCAN
Longitude of Observation Point for 89A	Ｒ４				
Latitude of Observation Point for 89B	Ｒ４	ＬＬ	AMTK_getLatLon()	AM2_LATLON_89B	486 x SCAN
Longitude of Observation Point for 89B	Ｒ４				
* 緯度(低周波平均)		ＬＬ	AMTK_getLatLon()	AM2_LATLON_LO	243 x SCAN
* 経度(低周波平均)					
* 緯度(6G)		ＬＬ	AMTK_getLatLon()	AM2_LATLON_06	243 x SCAN
* 経度(6G)					
* 緯度(7G)		ＬＬ	AMTK_getLatLon()	AM2_LATLON_07	243 x SCAN
* 経度(7G)					
* 緯度(10G)		ＬＬ	AMTK_getLatLon()	AM2_LATLON_10	243 x SCAN
* 経度(10G)					
* 緯度(18G)		ＬＬ	AMTK_getLatLon()	AM2_LATLON_18	243 x SCAN
* 経度(18G)					
* 緯度(23G)		ＬＬ	AMTK_getLatLon()	AM2_LATLON_23	243 x SCAN
* 経度(23G)					
* 緯度(36G)		ＬＬ	AMTK_getLatLon()	AM2_LATLON_36	243 x SCAN
* 経度(36G)					
Brightness Temperature (6.9GHz,H)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB06H	243 x SCAN
Brightness Temperature (6.9GHz,V)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB06V	243 x SCAN
Brightness Temperature (7.3GHz,H)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB07H	243 x SCAN
Brightness Temperature (7.3GHz,V)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB07V	243 x SCAN
Brightness Temperature (10.7GHz,H)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB10H	243 x SCAN
Brightness Temperature (10.7GHz,V)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB10V	243 x SCAN
Brightness Temperature (18.7GHz,H)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB18H	243 x SCAN
Brightness Temperature (18.7GHz,V)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB18V	243 x SCAN
Brightness Temperature (23.8GHz,H)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB23H	243 x SCAN
Brightness Temperature (23.8GHz,V)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB23V	243 x SCAN
Brightness Temperature (36.5GHz,H)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB36H	243 x SCAN
Brightness Temperature (36.5GHz,V)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB36V	243 x SCAN
Brightness Temperature (89.0GHz-A,H)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB89AH	486 x SCAN
Brightness Temperature (89.0GHz-A,V)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB89AV	486 x SCAN
Brightness Temperature (89.0GHz-B,H)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB89BH	486 x SCAN
Brightness Temperature (89.0GHz-B,V)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB89BV	486 x SCAN
Pixel Data Quality 6 to 36	Ｕ１	Ｉ４	AMTK_get_SwathInt()	AM2_PIX_QUAL_LO	486 x SCAN
Pixel Data Quality 89	Ｕ１	Ｉ４	AMTK_get_SwathInt()	AM2_PIX_QUAL_HI	486 x SCAN
Land_Ocean Flag 6 to 36	Ｕ１	Ｉ４	AMTK_get_SwathInt()	AM2_LOF_LO	243 x SCAN x 6
Land_Ocean Flag 89	Ｕ１	Ｉ４	AMTK_get_SwathInt()	AM2_LOF_HI	486 x SCAN x 2
Earth Incidence	Ｉ２	Ｒ４	AMTK_get_SwathFloat()	AM2_EARTH_INC	243 x SCAN
Earth Azimuth	Ｉ２	Ｒ４	AMTK_get_SwathFloat()	AM2_EARTH_AZ	243 x SCAN

Ｉ４:4byte 符号付き整数型、Ｉ２:2byte 符号付き整数型、Ｕ２:2byte 符号なし整数型、Ｕ１:1byte 符号なし整数型、
Ｒ８:8byte 浮動小数点型、Ｒ４:4byte 浮動小数点型、ＴＭ:時刻構造体、ＬＬ:緯度経度構造体

表 11　サンプルプログラムで取り扱った L1R データ一覧

HDF5 データセット名 *は抽出値のため非格納	HDF5 格納型	AMTK 出力型	AMTK アクセス関数	AMTK アクセスラベル	サイズ
Scan Time	Ｒ８	ＴＭ	AMTK_getScanTime()		SCAN
Latitude of Observation Point for 89A	Ｒ４	ＬＬ	AMTK_getLatLon()	AM2_LATLON_RS_89A	486 x SCAN
Longitude of Observation Point for 89A	Ｒ４				
Latitude of Observation Point for 89B	Ｒ４	ＬＬ	AMTK_getLatLon()	AM2_LATLON_RS_89B	486 x SCAN
Longitude of Observation Point for 89B	Ｒ４				
* 緯度(低周波用)		ＬＬ	AMTK_getLatLon()	AM2_LATLON_RS_LO	243 x SCAN
* 経度(低周波用)					
Brightness Temperature (res06,6.9GHz,H)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_RES06_TB06H	243 x SCAN
Brightness Temperature (res06,6.9GHz,V)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_RES06_TB06V	243 x SCAN
Brightness Temperature (res06,7.3GHz,H)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_RES06_TB07H	243 x SCAN
Brightness Temperature (res06,7.3GHz,V)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_RES06_TB07V	243 x SCAN
Brightness Temperature (res10,10.7GHz,H)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_RES10_TB10H	243 x SCAN
Brightness Temperature (res10,10.7GHz,V)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_RES10_TB10V	243 x SCAN
Brightness Temperature (res23,18.7GHz,H)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_RES23_TB18H	243 x SCAN
Brightness Temperature (res23,18.7GHz,V)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_RES23_TB18V	243 x SCAN
Brightness Temperature (res23,23.8GHz,H)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_RES23_TB23H	243 x SCAN
Brightness Temperature (res23,23.8GHz,V)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_RES23_TB23V	243 x SCAN
Brightness Temperature (res36,36.5GHz,H)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_RES36_TB36H	243 x SCAN
Brightness Temperature (res36,36.5GHz,V)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_RES36_TB36V	243 x SCAN
Brightness Temperature (res36,89.0GHz,H)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_RES36_TB89H	243 x SCAN
Brightness Temperature (res36,89.0GHz,V)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_RES36_TB89V	243 x SCAN
Brightness Temperature (original,89GHz-A,H)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB89AH	486 x SCAN
Brightness Temperature (original,89GHz-A,V)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB89AV	486 x SCAN
Brightness Temperature (original,89GHz-B,H)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB89BH	486 x SCAN
Brightness Temperature (original,89GHz-B,V)	Ｕ２	Ｒ４	AMTK_get_SwathFloat()	AM2_TB89BV	486 x SCAN
Pixel Data Quality 6 to 36	Ｕ１	Ｉ４	AMTK_get_SwathInt()	AM2_PIX_QUAL_LO	486 x SCAN
Pixel Data Quality 89	Ｕ１	Ｉ４	AMTK_get_SwathInt()	AM2_PIX_QUAL_HI	486 x SCAN
Land_Ocean Flag 6 to 36	Ｕ１	Ｉ４	AMTK_get_SwathInt()	AM2_LOF_RES_LO	243 x SCAN x 4
Land_Ocean Flag 89	Ｕ１	Ｉ４	AMTK_get_SwathInt()	AM2_LOF_RES_HI	486 x SCAN x 2
Earth Incidence	Ｉ２	Ｒ４	AMTK_get_SwathFloat()	AM2_EARTH_INC	243 x SCAN
Earth Azimuth	Ｉ２	Ｒ４	AMTK_get_SwathFloat()	AM2_EARTH_AZ	243 x SCAN

Ｉ４:4byte 符号付き整数型、Ｉ２:2byte 符号付き整数型、Ｕ２:2byte 符号なし整数型、Ｕ１:1byte 符号なし整数型、
Ｒ８:8byte 浮動小数点型、Ｒ４:4byte 浮動小数点型、ＴＭ:時刻構造体、ＬＬ:緯度経度構造体

表 12　サンプルプログラムで取り扱った L2 低解像度データ一覧

HDF5 データセット名	HDF5 格納型	AMTK 出力型	AMTK アクセス関数	AMTK アクセスラベル	サイズ
Scan Time	Ｒ８	ＴＭ	AMTK_getScanTime()		SCAN
Latitude of Observation Point	Ｒ４	ＬＬ	AMTK_getLatLon()	AM2_LATLON_L2_LO	243 x SCAN
Longitude of Observation Point	Ｒ４				
Geophysical Data	Ｉ２	Ｒ４	AMTK_get_SwathFloat()	AM2_SWATH_GEOA (*1)	LAYER x 243 x SCAN
Pixel Data Quality	Ｕ１	Ｕ１	AMTK_get_SwathUChar()	AM2_PIX_QUAL	243 x SCAN x LAYER

Ｉ４:4byte 符号付き整数型、Ｉ２:2byte 符号付き整数型、Ｕ２:2byte 符号なし整数型、Ｕ１:1byte 符号なし整数型、
Ｒ８:8byte 浮動小数点型、Ｒ４:4byte 浮動小数点型、ＴＭ:時刻構造体、ＬＬ:緯度経度構造体
(*1)1 層目だけ取得する場合は「AM2_SWATH_GEO1」、2 層目だけ取得する場合は「AM2_SWATH_GEO2」

表 13　サンプルプログラムで取り扱った L2 高解像度データ一覧

HDF5 データセット名	HDF5 格納型	AMTK 出力型	AMTK アクセス関数	AMTK アクセスラベル	サイズ
Scan Time	Ｒ８	ＴＭ	AMTK_getScanTime()		SCAN
Latitude of Observation Point for 89A	Ｒ４	ＬＬ	AMTK_getLatLon()	AM2_LATLON_L2_89A	486 x SCAN
Longitude of Observation Point for 89A	Ｒ４				
Latitude of Observation Point for 89B	Ｒ４	ＬＬ	AMTK_getLatLon()	AM2_LATLON_L2_89B	486 x SCAN
Longitude of Observation Point for 89B	Ｒ４				
Geophysical Data for 89A	Ｉ２	Ｒ４	AMTK_get_SwathFloat()	AM2_SWATHA_GEOA (*1)	LAYER x 486 x SCAN
Geophysical Data for 89B	Ｉ２	Ｒ４	AMTK_get_SwathFloat()	AM2_SWATHB_GEOA (*2)	LAYER x 486 x SCAN
Pixel Data Quality for 89A	Ｕ１	Ｕ１	AMTK_get_SwathUChar()	AM2_PIX_QUAL_A	486 x SCAN x LAYER
Pixel Data Quality for 89B	Ｕ１	Ｕ１	AMTK_get_SwathUChar()	AM2_PIX_QUAL_B	486 x SCAN x LAYER

Ｉ４:4byte 符号付き整数型、Ｉ２:2byte 符号付き整数型、Ｕ２:2byte 符号なし整数型、Ｕ１:1byte 符号なし整数型、
Ｒ８:8byte 浮動小数点型、Ｒ４:4byte 浮動小数点型、ＴＭ:時刻構造体、ＬＬ:緯度経度構造体
(*1)1 層目だけ取得する場合は「AM2_SWATHA_GEO1」、2 層目だけ取得する場合は「AM2_SWATHA_GEO2」
(*2)1 層目だけ取得する場合は「AM2_SWATHB_GEO1」、2 層目だけ取得する場合は「AM2_SWATHB_GEO2」

表 14　サンプルプログラムで取り扱った L3 輝度温度データ一覧

HDF5 データセット名	HDF5 格納型	AMTK 出力型	AMTK アクセス関数	AMTK アクセスラベル	サイズ
Brightness Temperature (H)	Ｕ２	Ｒ４	AMTK_get_GridFloat()	AM2_GRID_TBH	(*1)
Brightness Temperature (V)	Ｕ２	Ｒ４	AMTK_get_GridFloat()	AM2_GRID_TBV	(*1)

Ｉ４:4byte 符号付き整数型、Ｉ２:2byte 符号付き整数型、Ｕ２:2byte 符号なし整数型、Ｕ１:1byte 符号なし整数型、
Ｒ８:8byte 浮動小数点型、Ｒ４:4byte 浮動小数点型、ＴＭ:時刻構造体、ＬＬ:緯度経度構造体
(*1)P.10「図 6 L3 グラニュール ID」参照。格納順は(経度または横)→(緯度または縦)。

表 15　サンプルプログラムで取り扱った L3 物理量データ一覧

HDF5 データセット名	HDF5 格納型	AMTK 出力型	AMTK アクセス関数	AMTK アクセスラベル	サイズ
Geophysical Data	Ｉ２	Ｒ４	AMTK_get_GridFloat()	AM2_GRID_GEOA (*2)	(*1) x LAYER

Ｉ４:4byte 符号付き整数型、Ｉ２:2byte 符号付き整数型、Ｕ２:2byte 符号なし整数型、Ｕ１:1byte 符号なし整数型、
Ｒ８:8byte 浮動小数点型、Ｒ４:4byte 浮動小数点型、ＴＭ:時刻構造体、ＬＬ:緯度経度構造体
(*1)P.10「図 6 L3 グラニュール ID」参照。格納順は(経度または横)→(緯度または縦)。
(*2)1 層目だけ取得する場合は「AM2_GRID_GEO1」、2 層目だけ取得する場合は「AM2_GRID_GEO2」

１０．AMTK 定数

表 16～表 17 に AMTK に定義されている定数を示します。

表 16　AMTK 定数

内容	定数名	値
高解像度ピクセル数	AM2_DEF_SNUM_HI	486
低解像度ピクセル数	AM2_DEF_SNUM_LO	243
2byte 符号あり整数欠損値	AM2_DEF_IMISS	-32768
2byte 符号なし整数欠損値	AM2_DEF_UIMISS	65535
1byte 符号なし整数欠損値	AM2_DEF_CMISS	255
浮動小数点数欠損値	AM2_DEF_RMISS	-9999.0

表 17　L3 解像度定数

地図投影	方向	定数名	値
等緯度経度(低解像度)	経度	AM2_DEF_L3L_EQ_X	1440
	緯度	AM2_DEF_L3L_EQ_Y	720
等緯度経度(高解像度)	経度	AM2_DEF_L3H_EQ_X	3600
	緯度	AM2_DEF_L3H_EQ_Y	1800
ポーラステレオ北 (低解像度、TB・SIC)	横	AM2_DEF_L3L_PN1_X	304
	縦	AM2_DEF_L3L_PN1_Y	448
ポーラステレオ北 (高解像度、TB・SIC)	横	AM2_DEF_L3H_PN1_X	760
	縦	AM2_DEF_L3H_PN1_Y	1120
ポーラステレオ南 (低解像度、TB・SIC)	横	AM2_DEF_L3L_PS_X	316
	縦	AM2_DEF_L3L_PS_Y	332
ポーラステレオ南 (高解像度、TB・SIC)	横	AM2_DEF_L3H_PS_X	790
	縦	AM2_DEF_L3H_PS_Y	830
ポーラステレオ北 (低解像度、SND)	横	AM2_DEF_L3L_PN2_X	432
	縦	AM2_DEF_L3L_PN2_Y	574
ポーラステレオ北 (高解像度、SND)	横	AM2_DEF_L3H_PN2_X	1080
	縦	AM2_DEF_L3H_PN2_Y	1435